Delicate Nets, Faint Recollections:

A Study of Partially Connected Associative Network Memories

Jay T Buckingham

PhD
University of Edinburgh
1991

Declaration

I have composed this thesis myself and it reports original research that has been conducted by myself unless otherwise indicated.

Edinburgh, 25 June 1991

Jay Buckingham

Acknowledgements

I am grateful to David Willshaw for his constant support, guidance and confidence in me. He always knew the right time for encouragement and the right time for 'gentle prodding'. Academically, with his combination of mathematical rigour and experience in neuroscience he has steered me down a path of research that is perfect for me. It is no exageration to say that he has made this thesis possible. Pausing to look back, I am amazed at how much I have learned working with him. To David, thank you.

I was very fortuanate to be in good company in David's group. The many late night discussions with Marcus Frean and Peter Dayan at David's beat-up and sorely missed blackboard or in front of his fire on bleak winter nights played an important part in peeling layers of naiveté from my ideas. The entire group was very supportive throughout the development of this thesis. David, Marcus, Peter, and Toby Tyrrell painstakingly read drafts and made invaluable suggestions. And the teamwork demonstrated by David, Peter and Marcus on a warm summer morning in June was truly inspiring.

Many other people have influenced the eventual shape of this work. I would like to thank Henry Thompson for inviting me to Edinburgh in the first place and keeping me on the straight and narrow with the Faculty. Thankyou to Mike Malloch for getting the Edinburgh Connectionist Group together and especially for introducing me to David. Graham Rule actually volunteered to read the thesis (above and beyond the call of duty) and made very helpful comments on the prose. I really appreciate it. I had a number of fruitful discussion with Tony Gardner-Medwin about threshold setting in sparsely connected nets that definitely helped sharpen my thinking.

A special thankyou to Sally Wassel for sound advice.

My deepest thanks go out to Judy, Diane, Lou and Terri for friendship, love and support through the years.

The Medical Research Council provided generous financial support for this work, for which I am grateful.

Abstract

This thesis explores partially connected networks in associative memory tasks.

The storage capacity of the fully connected associative net is reviewed. Simulation results demonstrate that the performance of this architecture was worse than that predicted by earlier analyses. New analysis is presented that accurately describes the behaviour of this net.

A characterization of the storage capacity and information efficiency of the partially connected associative net is presented. In partially connected nets, one of the key problems is how to set the unit thresholds in such a way that the units which should not fire are quiet and those that should fire do so. New thresholding strategies are developed which enable this architecture to function well as an associative content-addressable memory. An important result is that the best thresholding strategies are functions of the firing history of an output unit and the number of active inputs impinging on it.

Analysis is presented that predicts the behaviour of single and multi-layer partially connected nets using the best thresholding strategy. It shows that for a large range of parameters values the information efficiency of the partially connected net is greater than that of both the fully connected associative net and simple competitive nets.

Contents

Chapter 1

Introduction

This thesis characterizes the functionality of a partially connected neural network in associative and content-addressable memory tasks. The motivation for the work presented comes primarily from studies on the structure and function of the hippocampal formation, a mammalian brain region critical for declarative memory. The architecture of this region is quite regular and resembles that of the matrix memories first studied in the 1960s (Willshaw *et al.*, 1969). Since the hippocampus is involved in memory and its architecture looks like that of associative network memories, many have suggested that it functions as one.

The architecture studied in this thesis consists of sets of binary linear threshold units which represent primary neurons connected by partially connected matrices of weighted directed links. The investigations presented build on the work of Willshaw *et al.*(1969), Marr (1971) and Gardner-Medwin (1976) and characterize the capabilities and limitations of this architecture with respect to associative memory tasks.

These authors investigated the storage capacity of associative network architectures using analytic techniques. The computing resources now available enable these architectures to be simulated, which can help verify the analyses. Analysis and simulation provide powerful tools which complement each other in understanding the behaviour of complex systems. Simulations are employed extensively in this thesis to motivate and validate the statements made about

the architecture. For example, in the development of this thesis it was found that simulation results did not agree with the capacity statements made by Willshaw and by Marr and this motivated further investigations of the assumptions made. It was found that one of the assumptions concerning the activity of units was simplistic. This thesis presents extended analysis which accounts for the behaviour actually observed. Thus simulations motivated improved analysis.

In partially connected nets, one of the key problems is setting unit thresholds in such a way that the units which should not fire during recall are quiet, and those that should fire do so. Marr suggests that thresholds should be a function of the number of active input lines impinging on a unit as well as the usual weighted sum of the inputs, but he does not propose any particular thresholding functions. One of the contributions of this thesis is the principled development of threshold setting strategies for partially connected nets. A number of strategies with different motivations are suggested, and then defined explicitly so that they can be simulated and analyzed. The distributions of the amount of input reaching the output units that should and should not fire are functions of the number of active input lines. A thresholding strategy based on this works best overall with respect to minimizing a simple output error measure.

In order to put this work into its proper context, the capacity of partially connected nets using activity dependent thresholds is compared to that of the fully connected associative net, to progressive recall in a partially connected net and to the standard formulation of competitive learning nets. Programs are developed which implement each of these. This thesis provides a comparison of these various associative net formulations using both analytic and simulation techniques. The capacity and information efficiency of the partially connected net is found to be greater than that of standard competitive nets, but usually less than that of the fully connected associative net.

It is shown in this thesis that partially connected nets can function well as associative or content-addressable memories. The analysis presented describes the performance to be expected for any particular parameter settings. We find that performance is very sensitive to the fraction of units active in an input or output pattern, with best performance (as found by many authors) when the

fraction active is low. Performance is not too sensitive to the connection density, so long as it is above a critical value; if it goes below this level performance degrades rapidly.

These results may be useful in the design of VLSI circuits that implement associative network memories and to those working to understand brain structures like the hippocampus. In VLSI, units are generally cheap relative to the connections between them, so an understanding of the performance of partially connected nets is important. With respect to understanding the brain, this thesis predicts (as others have before) that the fraction of primary neurons simultaneously active in a population must be low for that population to function well as a memory. More importantly, it predicts that the threshold at which a primary unit fires is a function not only of the excitatory influences upon it but also of its firing history and the number of active inputs.

Plan of the Thesis

Chapter 2: On associative network memories

Chapter 2 characterizes the capabilities of standard associative network memories. Willshaw *et al.* developed the canonical model – a fully connected network with binary valued synapses which stores pairs of binary vectors. These networks are studied using analysis and simulation.

Chapter 3: Partially Connected Nets: Dendritic Sum Thresholds

Chapter 3 introduces the partially connected associative net. It discusses the problems that must be addressed in setting unit firing thresholds in the partially connected net which do not arise in the fully connected case. In the partially connected net, false negative errors are possible as well as false positives. The setting of the firing threshold affects the number and kind of output errors – if it is set low, most of the units which should fire will do so, but many others will as

well, if it is set high, the number of false positives will be lower, but many more false negatives are likely. Thus design decisions must be made with respect to threshold setting. That is, "what sort of output patterns are 'better' and what sort are 'worse'?" Appendix B presents a summary of some commonly used pattern quality measures.

Chapters 4: Activity Based Thresholds

The idea that the amount of inhibition required to set output unit firing thresholds is dependent on the level of activity impinging on them seems simple, but has important ramifications. Chapter 4 explores the recall capabilities of associative network memories using a number of thresholding strategies which exploit knowledge of input activity.

Chapter 5: Self-Organizing Nets

In the canonical associative memory task, an input vector is paired with a desired or target vector such that subsequent presentation of the input elicits the target. In many brain regions, it is unlikely that some desired target vector will be specified *a priori*. However, such a prespecified target is not usually required. In many applications it not only suffices, but is preferable, to develop a new representation of the input pattern. During the training process a representation of each input vector is developed as a pattern of firing of the output units. This representation will be used by later processing steps. A version of the associative memory task can be formulated in which this representation is used as the target vector; that is, after training, presentation of one of the input vectors should elicit the representation associated with it on the output units. Chapter 5 explores the functionality of a partially connected net which constructs its own representations of input patterns during training in the associative and content-addressable memory tasks.

Chapter 6: Applications of Activity Based Thresholding

This chapter reviews the results of the thesis and discusses their implications for hippocampal theories. Conditions under which partially connected nets perform well in the associative and content-addressable memory tasks are discussed. The relevance of these results to theories of the hippocampus is noted. An updated estimate of the recall performance of Marr's model of the hippocampus is calculated. The chapter concludes with a speculative calculation of the storage capacity of the hippocampus if it is viewed as a partially connected associative network memory structure.

Chapter 7: Conclusions

The results of this thesis are summarized and areas for further development of the work presented are suggested.

Chapter 2

On associative network memories

2.1 Introduction

This chapter is about associative nets. It begins by describing the associative memory task as it will be used in this thesis. Several workers have developed associative net architectures which perform this task with varying degrees of success and the basic features common to these architectures are outlined. The associative net which is the topic of this chapter was developed and characterized in the late 1960's (Willshaw *et al.*, 1969; Willshaw, 1971). It provides the foundation from which to understand most of the work presented in this thesis, so we devote some time to it. The architecture is discussed, a summary of the original analysis is presented, and an example is presented. The canonical parameter values for this example are calculated, and simulations are run using these parameters, but we find the performance is much worse than that predicted by the original analysis. New analysis is presented which more accurately describes the actual behaviour of the net.

2.2 The associative memory task

In the context of this thesis, the task for an associative memory is storing ordered pairs of *events* in such a way that presentation of the first event to the memory will elicit the second event from it. Such a memory is called *content-addressable* if presentation of part of one of these events or a noisy version of one of them will elicit the whole of the event with which it is paired (associated). A common special case is the content-addressable autoassociative net in which events are paired with themselves; the goal in this case is recall of a full event by presenting some (small) part of it[1]. This is referred to as *pattern completion*.

The events are represented as vectors of numbers. The semantics of these numbers with respect to other systems is not specified. The first event in a pair of events to be associated is often called the *input vector*, the second the *target*. In most of the systems we will consider, the events are vectors with binary-valued elements. The elements of the binary vectors will often be referred to as *bits*. The elements of the target vector that take the high state (1) are called *genuine* or *high* and the ones that take the low state (0) are called *low*. Unless stated otherwise, in this thesis the input and output vectors have binary valued elements such that the number of bits in state 1 is constant for all input vectors and (perhaps a different) constant for all output vectors. These bits are chosen randomly. When a vector is presented to the memory for recall, the memory produces a vector called the *output* vector.

Vectors presented to the memory for recall are called *cues*. Cues are often classified by their relationship to a stored input pattern. For convenience, we refer to a cue that is the same as[2] one of the stored input patterns as a *full cue*. A cue that is constructed from a stored input pattern by changing the state of some of the elements from 0 to 1 and optionally changing some of the elements from 1 to 0 is called a *noisy cue*; those elements changed from 0 to 1 are called *noise* or *spurious* bits. A cue that is constructed by changing some of the elements from 1 to 0 without adding any noise bits is called a *partial cue*.

[1]The term heteroassociative memory is sometimes used to denote the memories in which the first and the second events of the pair are distinct.

[2]Each element of the cue is equal to the corresponding element of a stored input pattern.

In a heteroassociative memory, recall takes place in a single step – presentation of a cue elicits an output. Content-addressable recall can also be performed in a single step in an autoassociative memory, but in this type of memory recall can also proceed over a number of steps. An initial cue can elicit an output pattern which can be used as the cue to elicit another output pattern and so on until the target is elicited. In one type of recall from autoassociative memories, an initial partial cue is used and output patterns with incrementally more genuine units active are elicited in subsequent steps. This is known as *progressive recall.*

It is possible that recall from these memories may not be perfect. The output vector may not be the same as the desired target vector. In vectors with binary valued elements, the errors can be classified into two types: bits that are 1 in the target can be 0 in the output (false negatives) and bits that are supposed to be 0 can be 1 (false positives). The false positives are called *spurious* bits.

2.2.1 Performance measures

When we ask how well any machine performs a task we need to define some appropriate measure for that performance. With respect to recall in the associative memory task, several measures have been formulated. Which measure is appropriate depends on the purpose of the memory.

Hamming distance is an intuitive measure of the difference between two binary valued vectors of the same dimension; it is the number of elements at which they differ. Many workers have used the expected or mean value of the hamming distance between a stored target vector and the output vector as a measure of recall performance. It has the very useful property that for some network memories it is possible to make analytic statements about it. Hamming distance treats false positive and false negative errors equally. Depending on the use to which the memory is put it may be important to assign different costs to them in a performance measure. Possible error or pattern quality measures are discussed in Appendix B. This thesis mostly uses hamming distance because it is the most intuitive of the common pattern quality measures.

In many applications there may be criteria that an output pattern must meet such that recall is considered successful or 'good'. In these cases we may want to know what fraction of the stored patterns can be recalled successfully. For example, recall may be considered good if an output pattern differs by only zero or one bits from the desired target.

2.3 Associative Nets: What are they?

An associative net is a computational device that maps input vectors to output vectors. The elements of the input and output vectors are represented by *units* that take state 0 or 1. The input units connect to the output units via weighted links, called *weights* or *synapses* (Figure 2.1). Nets in which the input and output units are distinct sets of units are known as feedforward nets, or sometimes, heteroassociative nets. Associative nets in which the input and output units are the same set are known as autoassociative nets. In some architectures, the nets are fully connected, that is, each output unit is connected to each input unit (Steinbuch, 1961; Willshaw *et al.*, 1969), while others employ partially connected nets (Marr, 1971; Gardner-Medwin, 1976) in which the number of incoming links into an output unit is less than the number of input units.

The architecture is inspired by a simple abstraction of a set of primary neurons in the central nervous system and the nomenclature used to describe them reflects this. Primary neurons (often pyramidal cells) are those whose outputs project to other brain regions. The synapses they make with other neurons are usually excitatory. By contrast, another large class of neurons is the local inhibitory interneurons. They usually project only to neurons in their neighborhood. Their synapses are inhibitory. Their postulated function is to set the thresholds of the primary neurons dynamically (Marr, 1971).

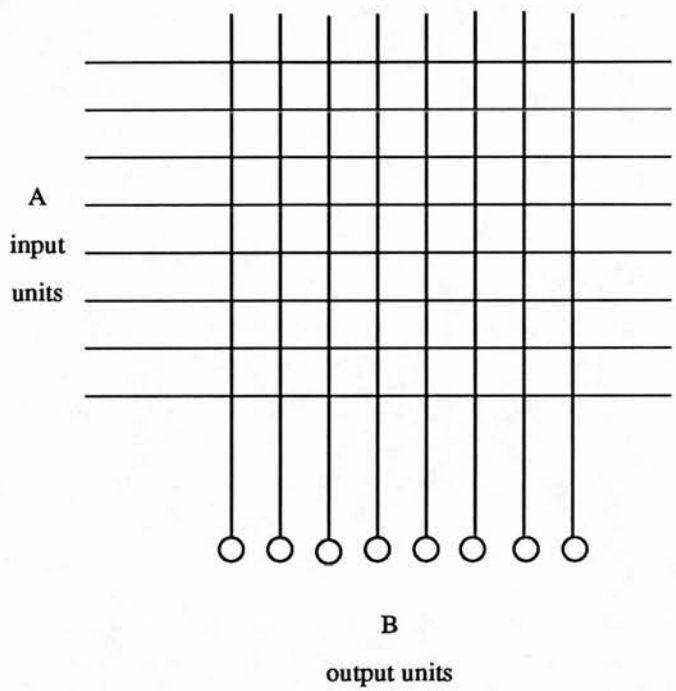

Figure 2.1: A standard schematic depiction of an associative net. The small circles represent the primary output units. The vertical lines represent the dendrites of these units. The horizontal lines represent the axons of the input units. Synapses are located at the intersection of the input axons and the output unit dendrites. Synapses with weight 1 will be indicated by filled circles in subsequent figures.

2.3.1 Notation

The various associative nets we will address share many architectural features. In order to discuss them conveniently let us define some notation which will be employed throughout this thesis.

The units that represent the elements of the vectors are said to be in layers or populations. The units that represent the input vectors make up layer A and those that represent the output layer B.

Parameters which define an architecture:

N_A	The number of input units
N_B	The number of output (primary) units
M_A	The number of input axons in state '1'
M_B	The number of output (primary) units in state '1'
α_A	The ratio M_A/N_A
α_B	The ratio M_B/N_B
S	The number of synapses onto an output unit
Z	The connection density, S/N_A

Notation describing the input, output and weight vectors:

x_i	$\in$	$\{0,1\}$	The state of input unit i
$\mathbf{x}$	$=$	$(x_1, x_2, \ldots, x_{N_A})$	The input state vector
y_i	$\in$	$\{0,1\}$	The state of output unit i
$\mathbf{y}$	$=$	$(y_1, y_2, \ldots, y_{N_B})$	The output state vector
w_{ij}	$\in$	$\{0,1\}$	The weight of the synapse from input unit j to output unit i
$\mathbf{w}_i$	$=$	$(w_{i1}, w_{i2}, \ldots, w_{iN_A})$	The weight vector into output unit i
$\mathbf{W}$	$=$	(w_{ij})	The weight matrix

A superscript is sometimes used to index the vectors.

Dynamic attributes of a network:

R	The number of pattern pairs stored in the net
$\hat{r}$	The expected number of patterns in which an output unit is active (which is $\alpha_B R$ when using random pattern sets)
r_i	The number of target patterns in which output unit i is active
ρ_i	The proportion of synapses on output unit i modified after storing the pattern pairs
ρ	The proportion of synapses in the net modified after storing the pattern pairs
$\hat{\rho}$	The expected proportion of synapses modified after storage
a_i^k	The number of active inputs impinging on output unit i when pattern k is presented to the net
d_i^k	The dendritic sum of output unit i when pattern k is presented to the net
m	The number of bits active (state 1) in a pattern
m_{cue}	The number of bits active in a cue
m_g	The number of genuine units active in a pattern (either input or output)
m_s	Number of spuriously active units in a pattern
δ_g	The number of genuine bits missing in a pattern

Functions:

ϕ	Performance
$\Delta(x^1, x^2)$	Hamming distance between vectors x^1 and x^2
$\cos(x^1, x^2)$	cosine of the angle between vectors x^1 and x^2, given by $\frac{x^1 \cdot x^2}{\|x^1\| \|x^2\|}$
$P(X = x)$	The probability that random variable $X = x$.
$F_X(x)$	$P(X \leq x)$ for the random variable X
$f(X = x)$	The frequency of occurrence of the event that random variable $X = x$, that is, the number of times $X = x$ when an experiment is performed divided by the number of times the experiment is performed.
$b(x; n, p)$	$P(X = x)$ for a binomial random variable X with parameters n and p.
$\text{poisson}(x; \lambda)$	$P(X = x)$ for a Poisson random variable X with parameter λ
$\bigvee_{i=1}^{n} x_i$	$x_1 \vee x_2 \vee \ldots \vee x_n$

2.4 The Associative Net

The canonical associative net was developed and analyzed by Willshaw and his colleagues (Willshaw *et al.*, 1969; Willshaw, 1971), from an initial interest in holographic memory. It is a fully connected net with binary valued synapses used for the storage of pairs of vectors with binary elements. It is of interest for a number of reasons. First, this model underlies a number of more complex network models so an understanding of it provides a foundation for understanding other related network architectures. Second, some neuroscientists feel that parts of the hippocampal formation implement associative networks, and third, analytic statements can be made about the behaviour of this network. This model will be discussed so frequently in this thesis we use the abbreviation WBL to abbreviate the 1969 paper by Willshaw, Buneman and Longuet-Higgins.

2.4.1 How does it work?

The association between patterns x^k and y^k is stored by modifying the weight matrix W.

Initially, $w_{ij} = 0$ for all i, j.

To store the set of pattern pairs $\{(x^k, y^k), k = 1, \ldots, R\}$ the weights are given by:

$$w_{ij} = \bigvee_{k=1}^{R} x_j^k y_i^k \tag{2.1}$$

That is, weights are set to 1 where there is conjoint activity on the input and output units. This is analogous to the hypothesis due to Hebb (1949) that the conjunction of pre- and post-synaptic activity at a synapse on a nerve cell can lead to an increase in the efficacy of the synapse.

In this model weights never decay.

To recall pattern k:
Let

$$d_i^k = \sum_{j=1}^{N_A} w_{ij} x_j^k$$

be the *dendritic sum* (or weighted sum of the inputs) impinging on unit i and

$$a^k = \sum_{j=1}^{N_A} x_j^k$$

be the number of active input lines, or simply the *input activity*. Then

$$y_i^k = \begin{cases} 1 & \text{if } d_i^k = a^k \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

That is, an output unit fires if its dendritic sum is equal to the number of active input lines.

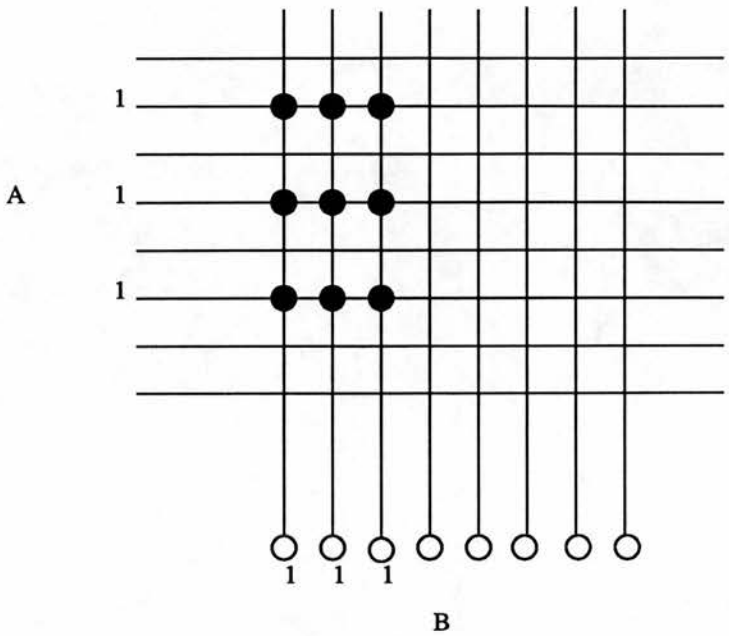

Figure 2.2: Associative net after storing pattern pair 1. The 1's next to the input and output lines denote the active units in the input and output pattern respectively. The filled circles represent weights with value 1.

Example:
Consider two eight bit pattern pairs:

$$x^1 = (0\,1\,0\,1\,0\,1\,0\,0) \qquad y^1 = (1\,1\,1\,0\,0\,0\,0\,0)$$
$$x^2 = (0\,0\,0\,0\,0\,1\,1\,1) \qquad y^2 = (0\,0\,1\,0\,0\,0\,1\,1)$$

After storing pattern pair 1 the weight matrix looks like the one depicted in Figure 2.2.

After storing both pattern pairs the weight matrix looks like the one depicted in Figure 2.3.

An important value to note is the proportion of synapses with weight '1' after storing the pattern pairs, ρ. In this example 17 of the synapses have weight 1 so $\rho = 17/64 = .266$

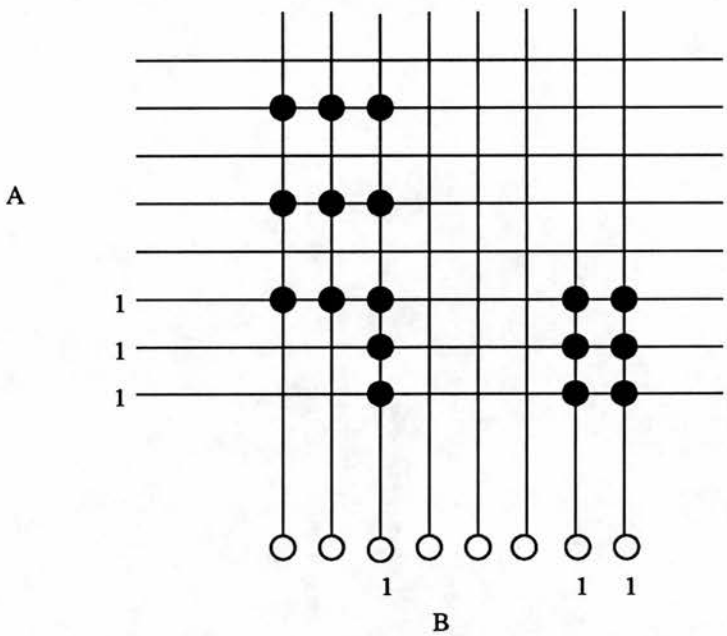

Figure 2.3: Associative net after storing the two pattern pairs (1 and 2). The 1's next to the input and output lines denote the active units in the input and output pattern respectively of pattern pair 2. The filled circles represent weights with value 1.

To recall $\mathbf{y}^1$ using pattern $\mathbf{x}^1$ as a cue, calculate a, the number of active input lines in $\mathbf{x}^1$, which comes to 3, and calculate d_i for each output unit. The vector of the d_i values is $\mathbf{d} = (3\,3\,3\,0\,0\,0\,0\,0)$. The output vector is obtained by performing the threshold operation on the dendritic sum vector based on a (equation2.2). This yields $\mathbf{y} = (1\,1\,1\,0\,0\,0\,0) = \mathbf{y}^1$ as desired.

2.4.2 Initial analysis

WBL were initially motivated by the question of whether holograms could be employed for associative and content addressable memory tasks. From the continuous holographic case they moved to a discrete formulation which developed into the associative net. In his thesis, Willshaw (1971) presents an analysis of the performance and information efficiency of this architecture.

The analysis contained in WBL and Willshaw (1971) is summarized here. They cast the problem as one of finding a set of parameters such that, during recall, the expected number of erroneous output bits is 1. Usually the parameters that define the net are stated and remain fixed while the number of pattern pairs stored varies. The analysis proceeds by calculating the expected proportion of synapses that are modified as a result of storing a number of such random pattern pairs and then calculating the probability that a low unit will incorrectly fire given that loading.

The net is parameterized by the number of input lines, N_A, the number of output lines, N_B, and the coding of the patterns to be stored. In this case, the input and output patterns have M_A and M_B active units respectively; the bits to be active in the patterns are chosen at random. A number R of pattern pairs is stored in the net by modifying the weight matrix W using the Hebbian rule expressed in equation 2.1.

The first step in the analysis is to develop an expression for the probability that a particular synapse w_{ij} has been modified during the storage process. The probability of any input line being active in a pattern is $\alpha_A = M_A/N_A$. Similarly for the output units the probability of any one of them being active in a target

pattern is $\alpha_B = M_B/N_B$. The probability of any particular synapse w_{ij} being modified during the storage of one pattern pair is then $\frac{M_A M_B}{N_A N_B}$ so the probability of it remaining unmodified (weight 0) is

$$1 - \frac{M_A M_B}{N_A N_B}$$

After storing R pattern pairs the probability of a synapse remaining unmodified is

$$\left(1 - \frac{M_A M_B}{N_A N_B}\right)^R$$

So the probability of it being modified is

$$\hat{\rho} = 1 - \left(1 - \frac{M_A M_B}{N_A N_B}\right)^R \tag{2.3}$$

$$\simeq 1 - \exp\left(-\frac{R M_A M_B}{N_A N_B}\right) \tag{2.4}$$

In evaluating how heavily the net can be loaded before performance degrades some performance measure is required. WBL considered recall performance to be good if the expected number of bits in error in an output pattern was less than or equal to one.

Consider recall using a stored input vector as the cue. The target vector has M_B units which should fire, and $N_B - M_B$ units which should remain quiet. We will use the letter g to subscript attributes such as dendritic sum d when that attribute applies only to the genuine units. Units which should remain quiet are called low units. If they do fire they are called spurious (false positives). The letter s is used to subscript the low/spurious units.

The synapses connecting the active input units to the genuine output units will have all been modified during training so the dendritic sum of these units is $d_g = M_A$. The synapses between the active input lines and the low units will have been modified with probability $\hat{\rho}$ so the dendritic sums of them are distributed binomially $b(M_A, \hat{\rho})$. The probability that the dendritic sum equals M_A (in which case the output unit will incorrectly be on) is then

$$b(x = M_A; M_A, \hat{\rho}) = \hat{\rho}^{M_A}$$

There are $N_B - M_B$ low units so the expected number of low units which incorrectly fire is $(N_B - M_B)\hat{\rho}^{M_A}$. The criteria for good recall in the WBL analysis is that this number be less than or equal to 1. They approximated this by

$$N_B \hat{\rho}^{M_A} = 1 \tag{2.5}$$

The number of pattern pairs which can be stored and still meet this criteria follows from equation 2.4

$$R \simeq -\frac{N_A N_B}{M_A M_B} \ln(1 - \hat{\rho}) \tag{2.6}$$

On the coding of the input representations, WBL use an information theoretic argument to show that sparse input patterns (with respect to the number of output units) make best use of this architecture. This argument is briefly outlined below.

The information required to represent one target pattern is $\log_2 \binom{N_B}{M_B}$ bits (Shannon, 1963) so after storing R patterns the information stored in the net is

$$I = R \log_2 \binom{N_B}{M_B}$$

This can be approximated using Stirling's approximation[3] by

$$I \simeq R M_B \log_2(N_B) \text{ bits}$$

From Equation 2.5

$$M_A = -\frac{\log_2(N_B)}{\log_2(\hat{\rho})}$$

So

$$\begin{aligned} I &\simeq -\frac{N_A N_B}{M_A M_B} \ln(1 - \hat{\rho}) M_B [-M_A \log_2(\hat{\rho})] \\ &\simeq N_A N_B \log_2(\hat{\rho}) \ln(1 - \hat{\rho}) \end{aligned}$$

[3]Stirling's approximation: $\log_2 \binom{n}{m} \simeq m \log_2(n)$, which holds for large n and small m/n

On Simulations

Simulations are used extensively in this thesis to motivate the investigations and validate the analysis. A set of computer programs was developed to support the simulation studies. This set consists of programs that generate data sets, programs that simulate the network models under study, and programs that analyze the data generated by the simulation programs. Many standard UNIX utilities are also exploited for data analysis.

The software was developed using standard structured design and programming techniques which facilitated modification and testing. It was developed in a modular fashion and implemented in C. All important functionality such as the generation of random patterns and the thresholding strategies to be discussed later were implemented as C functions. Initially these functions were imbedded in simple test harness programs for testing. Only after function testing were they integrated into the larger programs. The functions were grouped into source files based on the class of operation they perform. For example, the different threshold setting functions are each in separate source files so that their corresponding object files could be separately linked into an executable simulator. The compilation and linkage were maintained with the UNIX 'make' utility.

Several examples will serve to illustrate the testing techniques employed. Consider the sets of random pattern used as input and output patterns. One program generates random pattern sets. In order to test it, another program reads the representation of the pattern set (stored on disk) and performs several tests on it: (1) each pattern is checked to make sure it has exactly M bits on out of N, (2) the pairwise overlap of each pair of patterns in the set is calculated and the frequencies of the overlap values are printed, and (3) the number of times each element of the pattern takes value 1 (one) across the set is calculated (referred to as *unit usage*) and the frequencies of these values are printed. These tests show whether the patterns meet the specification of having exactly M bits on out of N and whether they were generated by a random process by comparing the overlap and unit usage frequencies to those expected of random bit vectors. Also consider a threshold setting function. This function takes a dendritic sum vector, a vector of the number of active inputs impinging on each output unit, a unit usage vector, and the desired number of active output units (along with scalars that specify the lengths of vectors) and returns a vector of the states of the output units after thresholding. Most of the threshold setting techniques vary the output unit thresholds in some systematic way until some criterion is reached, for instance, the desired number of output units are firing. This is usually implemented in a loop. During testing, the values of all relevant variables are printed for each loop cycle and either carefully examined or analysis performed on them to make sure they are behaving as required.

The results of the simulations form a key part of this thesis, but the implementation is not the topic of study so in the text simulation parameters are stated, but not implementation detail.

For fixed N and M, I is maximized at $\hat{\rho} = 1/2$, and therefore

$$M_A = \log_2(N_B)$$

2.4.3 Simulation and extended analysis

WBL's results were obtained via analysis. Are they borne out by simulation? This section compares the expected output error from analysis with the mean output error from simulations for an example net. The simulation results are much worse than that predicted by the WBL analysis which motivates a closer study of the factors underlying the actual behaviour of the net. New analysis is presented that does describe the actual behaviour observed.

Parameter values of the example

Let us consider the properties desired of an example net. First, we want a network as large as feasible with respect to our computing resource so as to minimize the effects of a finite sized net. Second, we want the coding of the patterns to be sparse but not logarithmically so since we will also be testing performance on recall of partial patterns and Willshaw (1971) notes that M must be greater than $\log_2(N)$ in order for this to work well.

Let $N_A = 8000$, $N = N_B = 1024$ and $M_A = 240$, $M = M_B = 30$.

Expected capacity of the example net

The value of $\hat{\rho}$ such that the expected number of output bits in error is one is:

$$\begin{aligned} \hat{\rho} &= \exp\left(-\frac{\ln(N_B)}{M_A}\right) \\ &= \exp\left(-\frac{\ln(1024)}{240}\right) \\ &= .9715 \end{aligned}$$

And the number of patterns which gives rise to that loading is

$$R = -\frac{8000 \cdot 1024}{240 \cdot 30} \ln(1 - .9715) = 4049$$

Comparison of expected and actual output error

Pattern sets with 240 (random) units on out of 8000 and 30 (random) units on out of 1024 were generated and used as the input patterns and target patterns respectively[4]. The associative net with the above parameter values was simulated. R pattern pairs were stored in the net ($R = 200, 400, \ldots 4800$), and then each input pattern was presented for recall. The simulator generated an output pattern, and a recall information structure holding data regarding the number of genuine and spurious bits in the recall cue and output pattern. This structure was suitable for later analysis.

Output error is defined as the hamming distance between the output pattern elicited and the target vector. For each simulation run, the mean output error is plotted in Figure 2.4 along with the expected value predicted by the analysis of Section 2.4.2. The actual performance is worse than predicted by WBL.

In most graphs, solid lines denote simulation results and dotted ones theoretical values. Some of the graphs become too cluttered with solid lines, in which case dotted lines are used for certain simulation results. This is explained and denoted in the figure legends.

Extended analysis

Let us take a closer look at the simulation runs where the net was loaded such that the expected output error was 1. For $R = 4000$, the results for one typical run are $\rho = .97$ and the mean and variance of the output error are 4.5 and 4.48

[4]Simulations are used extensively in this thesis to motivate the investigations and validate the analysis. Though the results are useful, the implementation is not the topic of study so in the text of the thesis simulation parameters are stated, but not implementation detail.

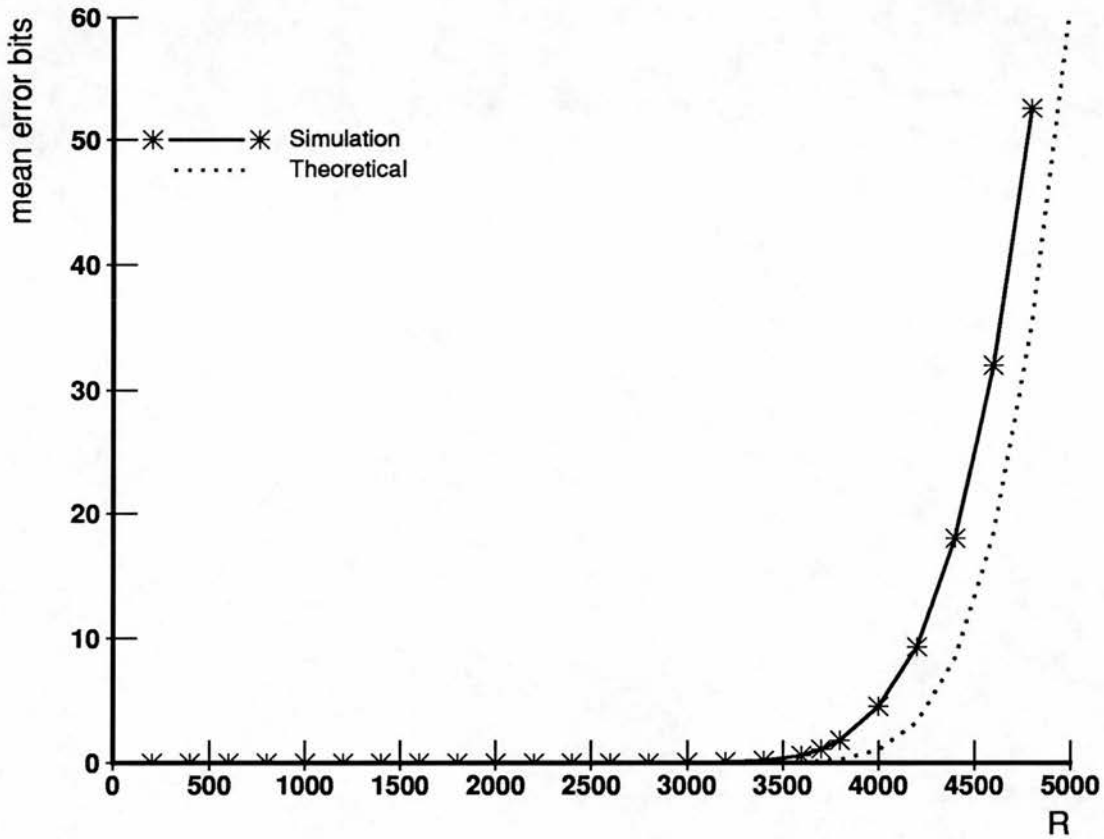

Figure 2.4: Performance of the associative net as a function of R, the number of patterns stored: simulation results and value predicted by WBL. Performance is the mean number of error bits in a recalled pattern (expected value for the theoretical prediction). Although the curves do not appear all that far apart, note the large difference in performance for any particular value of R above 4000.

respectively. Runs at $R = 4000$ were done for the other input and target patterns sets (10 sets in total). The mean output errors across these runs have mean 4.048 and standard deviation .236. The variances of the output error have mean 4.10 and standard deviation .215. The low standard deviations show that the results are quite repeatable. The WBL analysis predicts that the expected number of error bits is 1.

We can understand the discrepancy by looking at the distribution of the values of the dendritic sum of the low units, d_s. Figure 2.5 shows this frequency for the simulation runs noted and the binomial distribution parameterized by M_A, the number of active input units per pattern, and $\hat{\rho}$, the fraction of synapses modified after storage. The d_s frequency is much broader than the binomial which should have predicted it. Error bars showing one standard deviation of the d_s frequency across the 10 simulation runs are also plotted in this figure. Note that the standard deviation is consistently small.

The WBL analysis uses one value for the probability of modified synapses in the expression for the expected value of the number of spurious output units. This assumption is the cause of the discrepancy between the prediction of that analysis and the actual behaviour of the net. The parameter $\hat{\rho}$ is calculated for the entire net, but the fraction of synapses modified on an output unit varies from unit to unit. The frequency of ρ_i (from simulations) is shown in Figure 2.6. The WBL analysis is based on describing the d_s distribution with the binomial $b(M_A, \hat{\rho})$, which does not hold for every unit.

The variation in the values of ρ_i is a consequence of characteristics of the pattern sets. Though each target pattern has 30 bits on out of 1024, it is not guaranteed that across the entire pattern set each output unit will be on $\frac{30}{1024}R$ times. On average, for a given unit, the fraction of the number of patterns in which it fires is $\alpha_B = M_B/N_B$. The number of times an output unit fires in R patterns can be described with the binomial distribution $b(R, \alpha_B)$. The actual number of times an output unit is in the high state across the target patterns is denoted by r_i, often called the unit usage. Graphs comparing the actual frequencies of the unit usage in some of the pattern sets used with the binomial are shown in Appendix A.

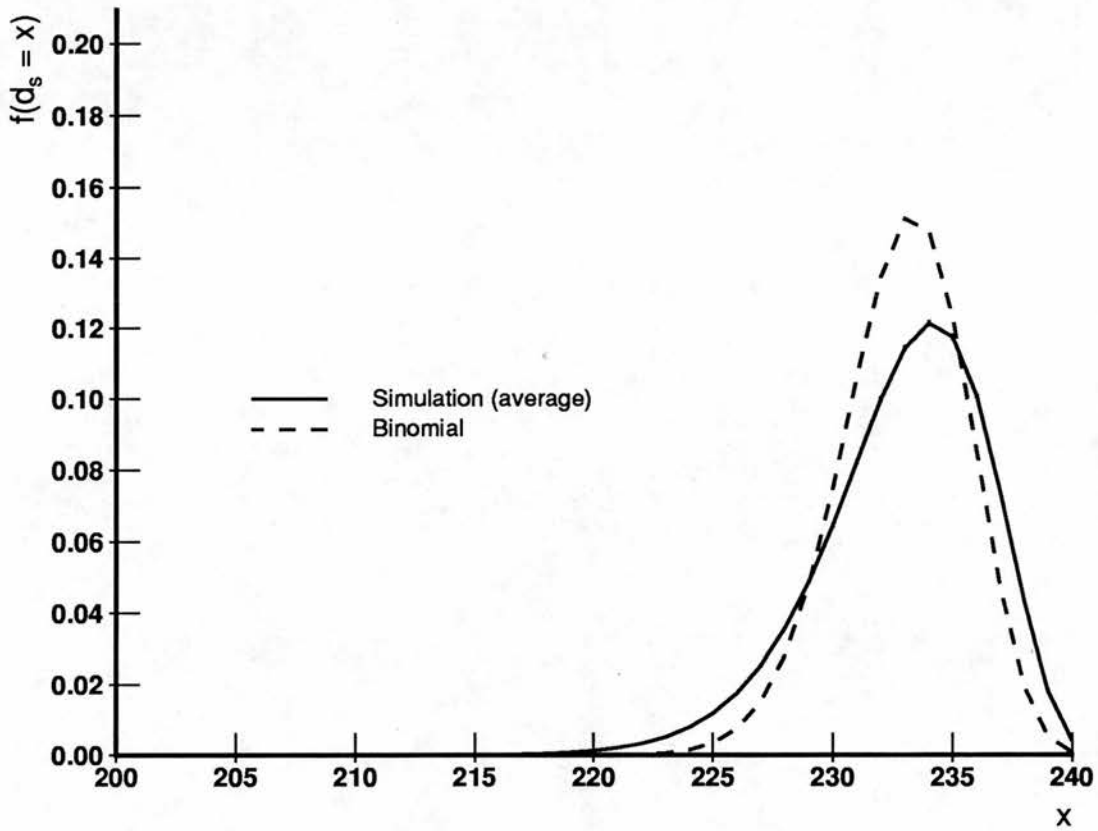

Figure 2.5: Frequency of $(d_s = x)$ in the simulation example compared with the binomial distribution with parameters $M = 240$ and $\rho = 0.970317$. Note that the d_s distribution is much broader than the binomial. Note too the very small error bars on the curve for the simulation data which mark one standard deviation across the simulation runs. $N_A = 8000$, $N_B = 1024$, $M_A = 240$, $M_B = 30$, $R = 4000$ and $\rho = 0.970317$.

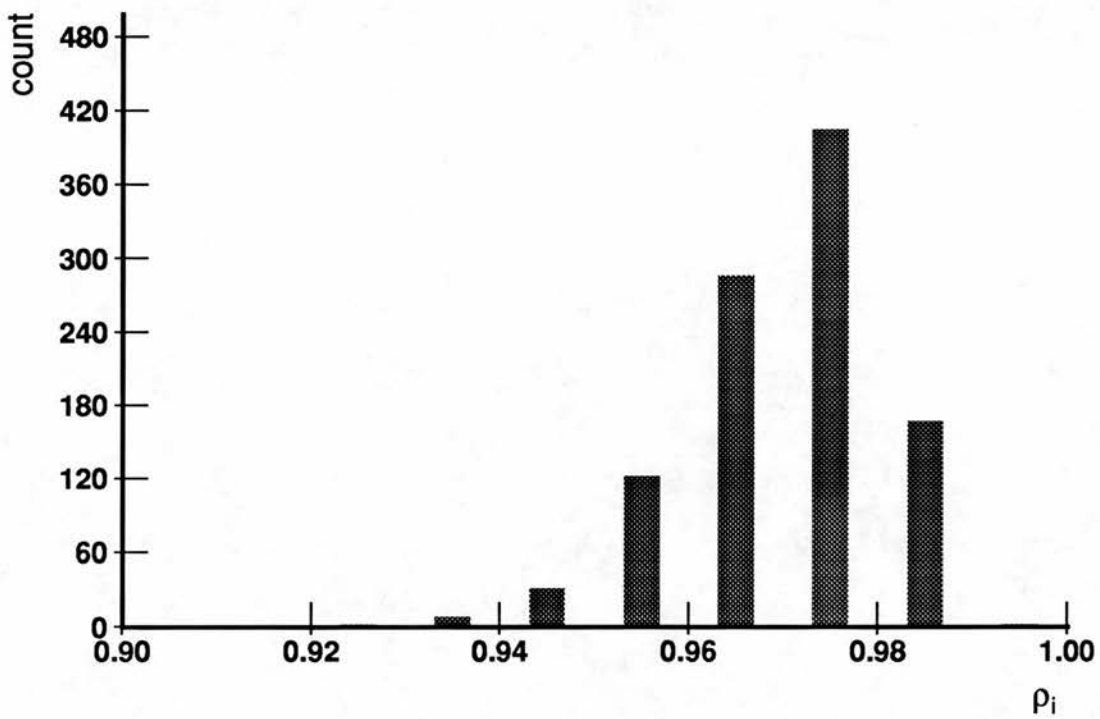

Figure 2.6: Histogram of ρ for the associative net example with $N_A = 8000$, $N_B = 1024$, $M_A = 240$, $M_B = 30$, $R = 4000$. The theoretical value of ρ at this value of R is 0.97 and mean and variance of these data are 0.97 and .0001 respectively.

If unit i is set in the high state r_i times then the chance of any synapse on it being modified is

$$\begin{aligned}\rho_i &= 1-(1-\alpha_A)^{r_i} \\ &\simeq 1-\exp(-r_i\alpha_A)\end{aligned}$$

This expression will arise so frequently that it is convenient to define the function

$$\rho(r) = 1-(1-\alpha_A)^r \tag{2.7}$$

where r is the number of target patterns in which an output unit fires.

Figure 2.7 shows ρ_i as a function of r_i: both the theoretical curve and a scattergram of the simulation data are plotted. The theoretical relationship describes the actual (r_i, ρ_i) data quite well.

Thus we have found the underlying cause of the discrepancy between the WBL prediction and the actual behaviour of the net: unit usage, r_i, comes from a binomial distribution instead of being constant as was assumed in the original analysis.

We can now better describe the d_s distribution by taking into account the distribution of the r_i. We are interested in $P(d_s = x)$, the probability that d_s takes some particular value x. The r_i are taken from the binomial distribution $b(R, \alpha_B)$ so

$$P(r = k) = b(k; R, \alpha_B) = \binom{R}{k} \alpha_B^k (1-\alpha_B)^{(R-k)}$$

The d_s are distributed binomially for any particular value of r:

$$\begin{aligned}P(d_s = x \mid r = k) &= b(x; M_A, \rho(k)) \\ &= \binom{M_A}{x} (\rho(k))^x (1-\rho(k))^{M_A - x}\end{aligned}$$

which gives

$$P(d_s = x) = \sum_{k=0}^{R} \binom{R}{k} \alpha_B^k (1-\alpha_B)^{R-k} \binom{M_A}{x} (\rho(k))^x (1-\rho(k))^{M_A - x} \tag{2.8}$$

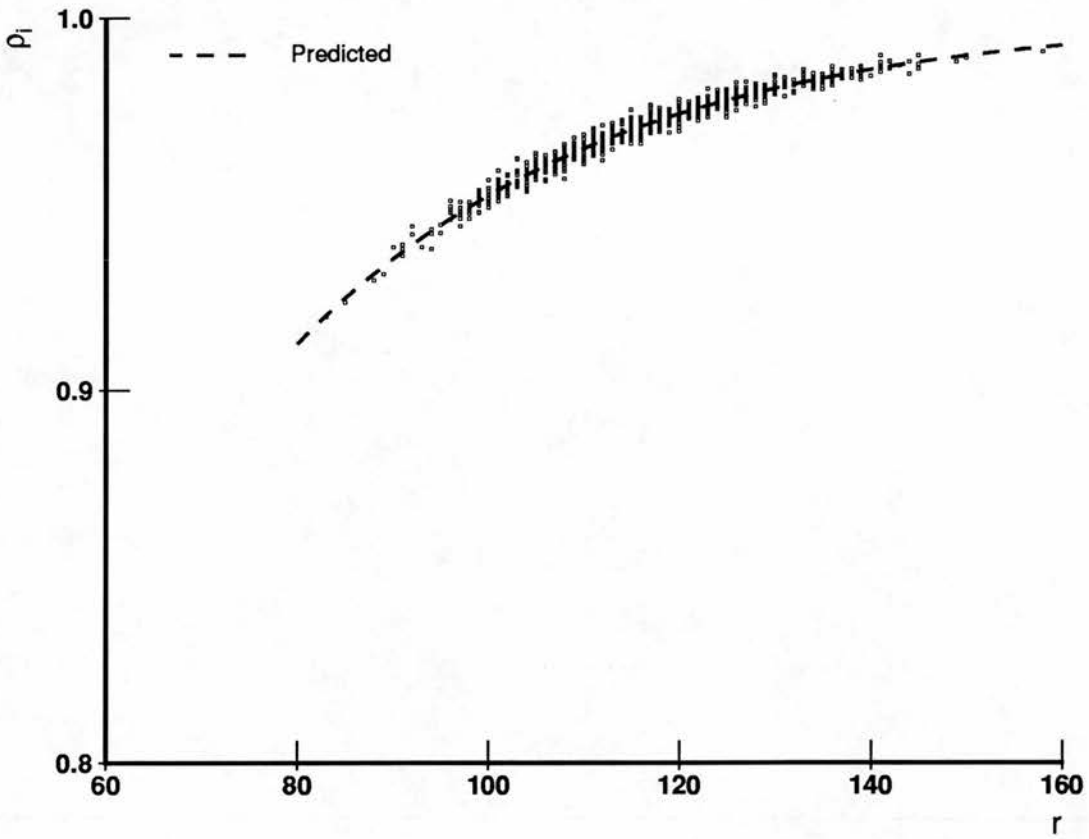

Figure 2.7: ρ_i as a function of τ_i. The line shows the theoretical relationship between them and the scattergram the simulation data. The theoretical relationship describes the data well. In the simulations and for the calculations, the parameter values of the example net are used, and $R = 4000$.

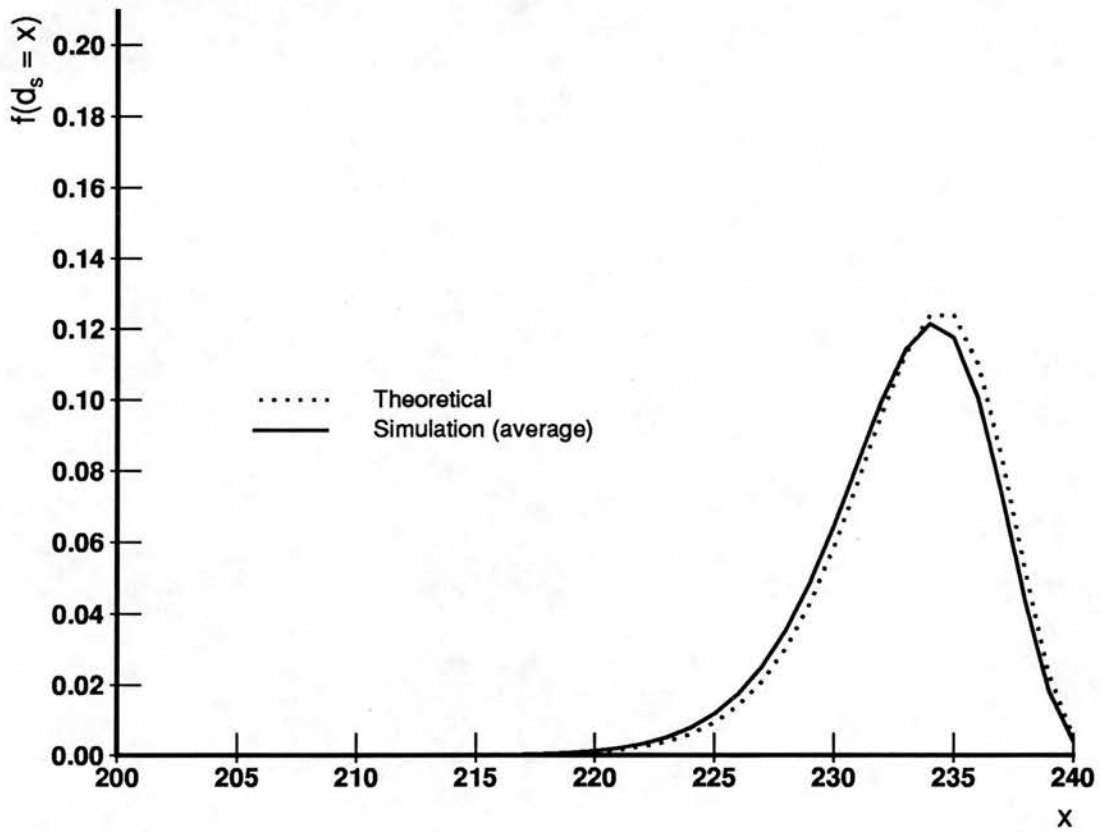

Figure 2.8: Probability $P(d_s = x)$ compared with the $f(d_s = x)$ in the simulation example. $R = 4000$.

In Figure 2.8 the d_s frequency from simulations (same curve as in Figure 2.5) is compared with $P(d_s = x)$ and the match is quite good.

To calculate the expected number of false positives in the associative net we need to know the probability that the dendritic sum for a low unit equals M_A. For the example net, using Equation 2.8 gives $P(d_s = 240) = 0.0054$, which is significantly greater than $\hat{p}^{M_B} = .970317^{240} = .00072$ used in the original analysis. The expected number of errors with this probability of spurious units is $(N_B - M_B)P(d_s = 240) = 5.34$ which is in accord with the mean number of spurious units observed in the simulation example.

Thus new analysis has been developed in this thesis that accurately describes the behaviour of the associative net.

Recall using partial cues: analysis and simulation

We now have a better understanding of the behaviour of the associative net when an input pattern of a stored pattern pair is presented for recall. One question that arises at this point is "How well does the associative net perform the content-addressable memory task?" The analysis of the previous section is extended to the case of recall from cues which are constructed from stored input patterns by turning off some fraction of the active bits. Simulations are performed, and the results show this new analysis to be in accord with the simulation results.

Let us begin by studying simulation results in order to gain an overall view of the behaviour of the net in the content-addressable memory task. Using the parameter set of our continuing example, a simulation run was done with $R = 3600$. At this loading the net defined by the canonical parameter values performs very well using stored input patterns as cues. After storage, recall was tested using partial cues. Let $g \in [0, 1]$ be the fraction of genuine bits that are present in the cue[5]; that is, $m_g = gM_A$. To construct a partial cue from a stored

[5]Note the differing uses of the letter g: as a subscript it denotes that the symbol subscripted applies to 'genuine' units only; as a parameter it denotes a real number. However, in both cases it has to do with 'genuine' units.

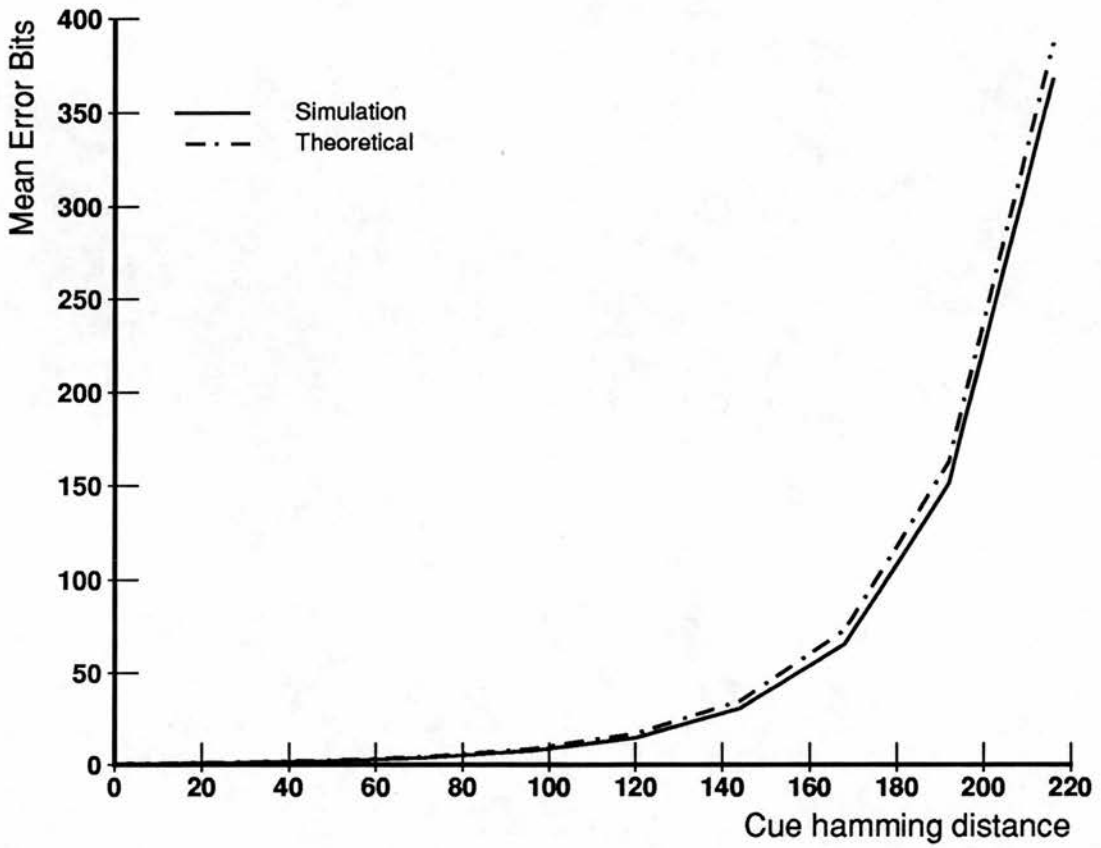

Figure 2.9: Performance of the associative net using partial patterns as cues: theoretical and simulation. The horizontal axis is the hamming distance between the recall cue and the stored input pattern. Performance is mean number of bits in error in the output patterns. $N_A = 8000$, $N_B = 1024$, $M_A = 240$, $M_B = 30$, $R = 3600$.

input pattern for a given value of g, active bits in the input pattern were turned off randomly until $m_{cue} = m_g = gM$, where m_{cue} is the number of bits active in a cue. For each pattern, cues were presented for $g = .1, .2, \ldots, 1.0$. The cue was presented to the net, an output pattern generated by the simulator, and the recall information data structure stored for later analysis. In Figure 2.9, mean output error is plotted as a function of the hamming distance between the cue and the stored pattern given, which is closely related to g. Not surprisingly, as the cue becomes more and more 'partial', performance degrades.

The analysis of recall errors using partial cues is a generalization of that for full

cues. Using partial cues the distribution of the dendritic sums for the low units is given by

$$P(d_s = x) = \sum_{k=0}^{R} \binom{R}{k} \alpha_B^k (1-\alpha_B)^{R-k} \binom{m_{cue}}{x} \rho(k)^x (1-\rho(k))^{m_{cue}-x}$$

Simulation results and expected error in Figure 2.9 show that the simulation data are in accord with the expectation.

Summary: An analysis has been presented of the performance of the associative net in the content-addressable memory task when presented with partial versions of stored input patterns as recall cues. The performance degrades with the hamming distance between the stored input pattern and the recall cue.

The other important case is recall from noisy cues. Recall performance using noisy cues in the canonical associative net is very poor since the simple thresholding strategy sets the firing threshold at the number of active bits in the cue. Willshaw (1971) treats this topic briefly. He considers the case where noisy cues are presented to the net and he assumes that the threshold of the output units can be set to the number of genuine bits in the cue. In practice, a net will not have information about the number of genuine bits in the cue available to it. In any event, the goal of this chapter is to study the canonical associative net with its simple threshold rule. More complex thresholding strategies are developed in Chapter 4.

2.4.4 On information efficiency

Information efficiency may be defined as the total amount of information (in bits) which can be recalled from the net divided by the number of bits of storage. Since each output pattern has the same number of active bits, they all 'contain' the same amount of information. This value, which we will call I_0, is given by

$$I_0 = \log_2 \binom{N_B}{M_B} = \log_2 \binom{1024}{30} = 191.67 \text{ bits}$$

(Shannon, 1963).

In order to calculate information efficiency, the number of pattern pairs that can be stored reliably in the net must be found. Define 'good recall' to be recall such that the mean number of error bits is less than or equal to 1 and let R_{good} be the maximum number of pattern pairs that can be stored in the net such that good recall is obtained. Since there are $N_A N_B$ weights in the net and each requires 1 bit of storage, the information efficiency (η) of the fully connected associative net is given by

$$\eta = \frac{R_{good} I_0}{N_A N_B}$$

Using full cues, the example net is able to store approximately 3600 pattern pairs and still deliver good recall. Thus the information efficiency of this net is $\eta = .084$. This does not seem very high compared to the limiting information efficiency figure of ln(2) stated in (Willshaw, 1971), but it is to be expected since the coding of the patterns is such that $M > \log_2(N)$. The WBL analysis predicted that 4049 pattern pairs could be stored with good recall. This would give $\eta = .095$, which is not much higher than that defined by using the actual value of $R_{good} = 3600$.

2.5 Summary

This chapter has focused on the canonical associative network as developed by Willshaw *et al.*. This network provides a foundation for the network architectures studied in the rest of this thesis so an understanding of how it works is important. Simulation results show that it indeed functions well as an associative memory. However, it does not function as well as predicted by the original analysis and this discrepancy motivated further study. The erroneous assumption is that each output unit will be active in the same number of target patterns; this is not the case for random pattern sets. The original analysis is extended by taking this into account in the expression for the probability of a spurious unit firing. The predictions made by the extended analysis of the expected number of output errors are found to match simulation results well.

The rest of this thesis is a study of partially connected associative nets. The analysis developed in this chapter will be adapted and extended to the partially

connected case in the remaining chapters.

Chapter 3

Partially Connected Nets: Dendritic Sum Thresholds

This and the following two chapters explore how the attributes of partially connected associative nets change as functions of the parameters of the net. Naturally, recall performance as a function of the number of patterns stored is of primary interest. An understanding of the influence of factors such as the fraction of modified synapses on an output unit, the number of active inputs impinging on a unit, and the dendritic sum of a unit, enables analytic statements to be made about the performance that can be achieved. This understanding also motivates the development of threshold setting schemes which yield good performance in practical settings.

These three chapters form the core of this thesis. They consider the *single layer* network architecture in which a set of input units projects to a set of output units. The projection is partial - each output unit is contacted by only a fraction of the input units. With respect to storing the association between input and target patterns there are two cases: the targets can be specified *a priori* or they can be developed by the net. Chapters 3 and 4 address the associative case, which is the case where target vectors are specified. Thresholding strategies that make use of only dendritic sums to determine which units do or do not fire are developed in Chapter 3 and those that additionally exploit information about the number of active inputs that impinge on an output unit are developed in Chapter 4. Building on this work, Chapter 5 discusses the self-organizing

case, in which target vectors for each input vector are determined during and by the training process. For each of these cases we will examine behaviour of the attributes when presented with full, partial, and noisy recall cues.

This chapter begins with a discussion of the architecture of a partially connected associative net that will be used to illustrate the ideas presented. Next, since the distributions of the dendritic sums of the high and low units underly recall performance, an analysis of these distributions is presented. Then threshold setting strategies based on the dendritic sums are developed and their performance is compared. The recall performance of most of them is not impressive. However, progressive recall using dendritic sum thresholds does deliver good performance. This method only applies to the autoassociative net. The poor performance of the dendritic sum based strategies for heteroassociative nets motivates the development of better thresholding strategies in Chapter 4.

3.1 Associative Case: Introduction

Here we consider a partially connected associative net. The input and output vectors are specified *a priori*. The elements of these vectors and the weights are binary valued. Training is performed using the usual associative net training regime[1] denoted in equation 2.1 (Willshaw *et al.*, 1969).

Setting the firing threshold for units in partially connected nets is not as straightforward as in the fully connected case since the values of the dendritic sums of both the low *and* the genuine units are probabilistic. We will study their distributions in this chapter.

[1]With the obvious difference that non-existent weights are not modified.

	A	B
N	8000	1024
M	240	30
α	0.03	0.029
S		5333
Z		.666

Table 3.1: Parameter values for the architecture of the single layer example net, referred to as the canonical parameter set.

3.2 Parameter values of an example net

The parameter values of the example net which will be most useful in our explorations need to be established. As in Chapter 2, we want as large a net as possible to give us as fine a grained view as possible of the dendritic sum distributions; M_A should be set high that the dendritic sums can take on a large number of different values. In the partially connected case, the connection density also affects the dendritic sums. Z should be set high enough so that the dendritic sums can still take on a large number of different values. Let $N_A = 8000$, $N = N_B = 1024$ and $M_A = 240$, $M = M_B = 30$ as in Chapter 2, primarily because a net defined by these parameter values can be simulated with the computational resources available. Z is set at $\frac{2}{3}$ which is large enough so that the dendritic sum distributions will still be reasonably fine-grained, but sufficiently less than 1.0 that the effects of partial connectivity can be investigated.

The parameter values which define this canonical net are collected in Table 3.1. This net will be used as an example to illustrate points being made in the analysis. Later in the chapter, simulation results are presented which show the sensitivity of recall performance to changes in these parameters.

Unless stated otherwise, simulations are performed using these parameter values[2].

[2]This is noted in most of the figure captions, but is omitted when it would be redundant.

In order to run simulations, input and target pattern sets are required. The pattern sets used in Chapter 2 are used here as well.

3.3 Characterizing the dendritic sum distributions

The recall performance of partially connected associative nets is readily understood in terms of the dendritic sum distributions of the low and high units. This section characterizes these distributions as a function of the parameters of the net and the nature of the recall cues, i.e. whether they are full, partial or noisy.

3.3.1 Recall using full cues

We first examine the distributions of the dendritic sums of the low and genuine units during recall using stored input patterns as recall cues. During storage, the synapses of the active units of a target pattern which received active inputs were all modified. There are M_A active bits in each input pattern and the contact probability is Z, so the dendritic sums on the genuine units are distributed $b(M_A, Z)$. For the low units, we might predict that their dendritic sums will be distributed $b(M_Z, Z\hat{\rho})$, but as in the fully connected case described in the previous chapter, the distribution is much broader than the binomial. In order to describe the distribution we must take into account the situation that the fraction of synapses ρ_i modified on output unit i varies with the number of patterns r_i in which it fires.

The situation is analogous to that analyzed in Chapter 2, except the connection density must also be accounted for. The r_i are distributed $b(R, \alpha_B)$ and the d_s are distributed $b(M_A, Z\rho(k))$ for any particular value of $r = k$, so

$$P(d_s = x) = \sum_{k=0}^{R} \binom{R}{k} \alpha_B^k (1-\alpha_B)^{R-k} \binom{M_A}{x} (Z\rho(k))^x (1 - Z\rho(k))^{M_A - x} \quad (3.1)$$

Figure 3.1 shows histograms of the number of occurrences of dendritic sum values for the low and genuine units in a network simulation using the canonical

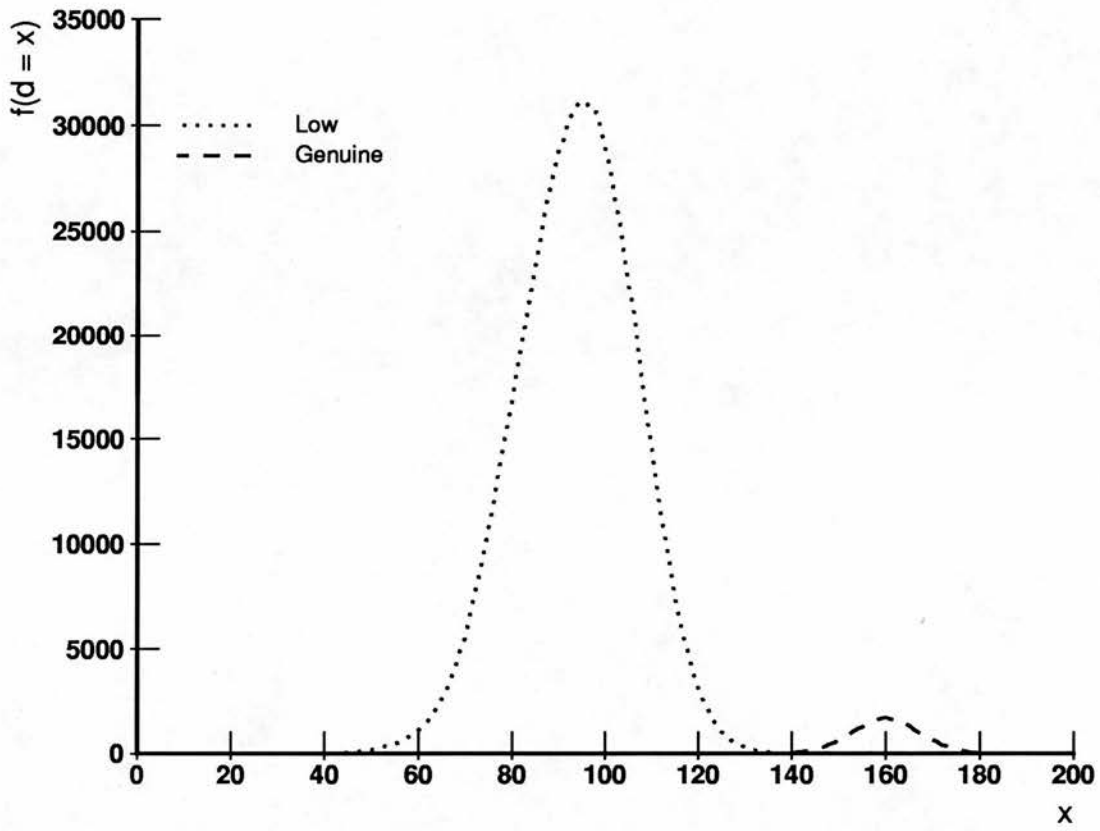

Figure 3.1: Histogram of the number of occurences of the d values for the low and genuine units. Network parameters appear in Table 3.1, R = 1000.

parameters in Table 3.1 after storage of 1000 pattern pairs. There are many more low units than genuine ones, as is clearly visible in the histogram. Figure 3.2 shows the frequencies for the low and genuine units averaged over twenty runs; the error bars indicate one standard deviation. The small standard deviations show that the frequencies are stable using the pattern sets constructed as described in Chapter 2. The frequency of the low units lies much to the left of that of the high units because at this loading the mean of the dendritic sum distribution of the low units is 94.4 and that of the high is 160. (At R = 1000, $\hat{p} = .59$.)

Figure 3.3 compares the results of the simulations to the predicted distributions for the genuine and low units respectively (given by $b(M_A, Z)$ and equation 3.1). The frequency for the genuine units is supposed to be binomial, and indeed the

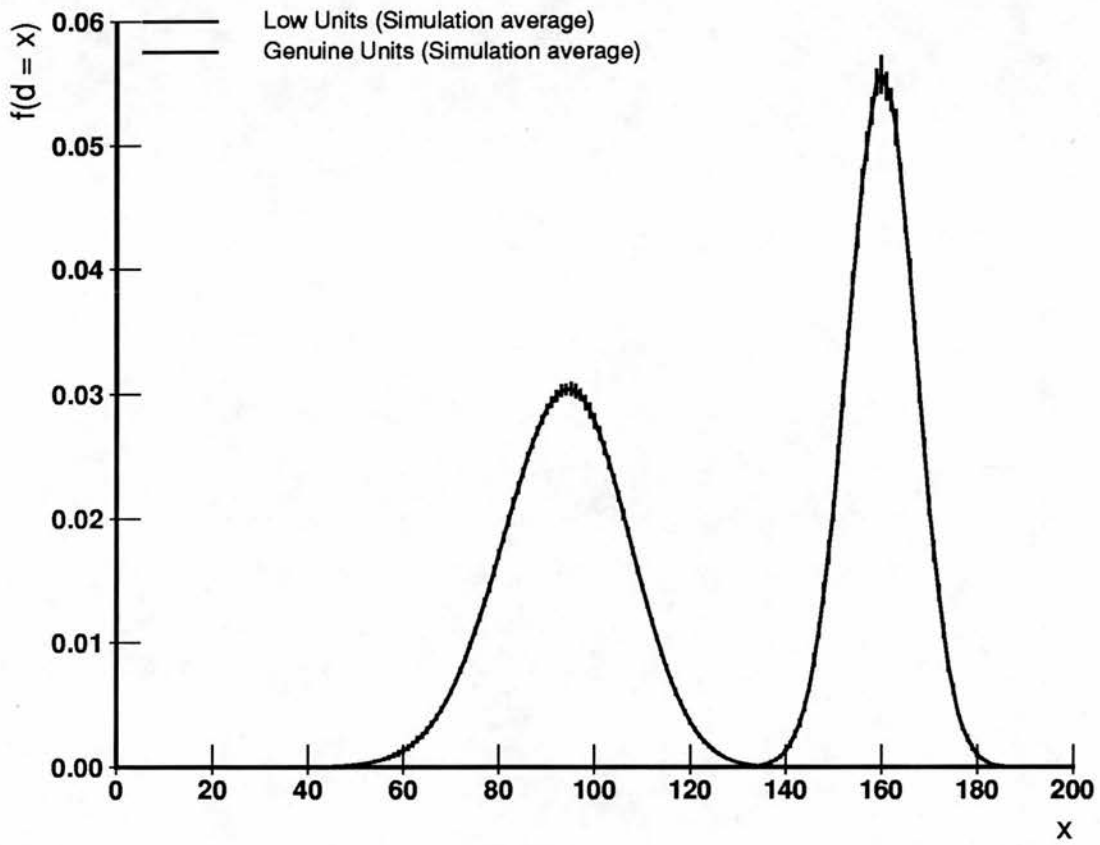

Figure 3.2: Dendritic sum frequency for low and genuine units averaged over 20 simulation runs with error bars marking one standard deviation. (Note that the error bars are small. Solid curves are used to plot both sets of data since dotted lines would obscure the small error bars.) Network parameters appear in Table 3.1, R = 1000.

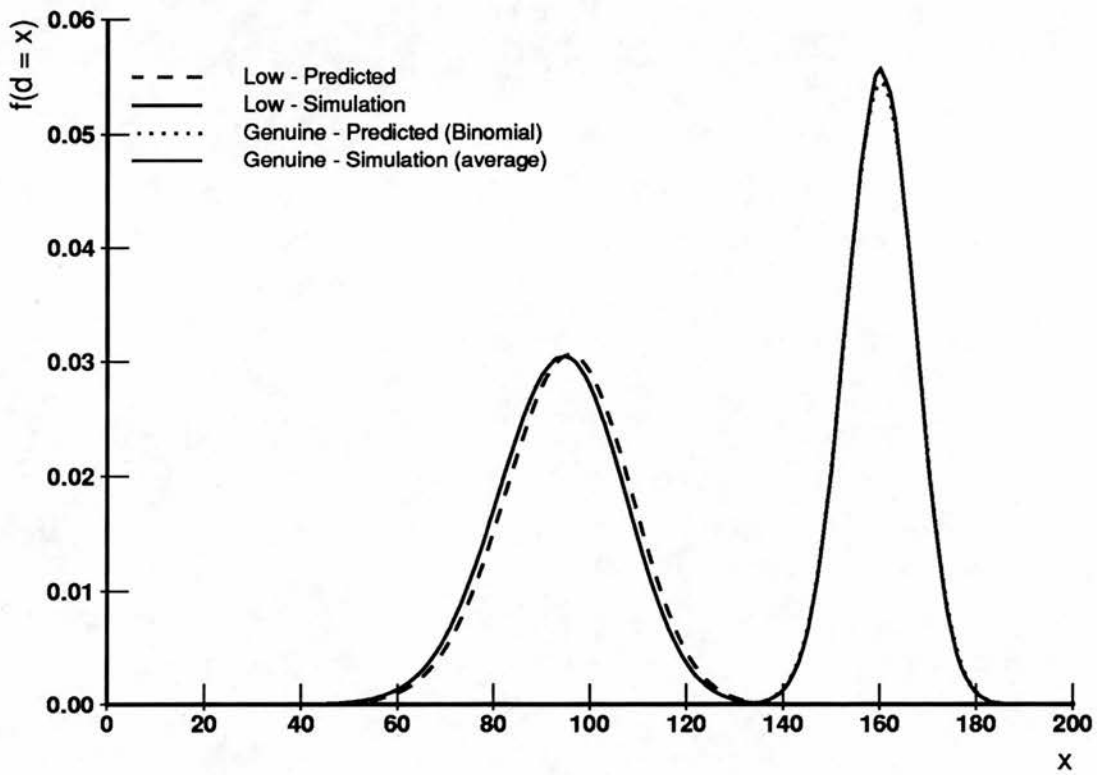

Figure 3.3: Dendritic sum frequency for low and genuine units averaged over 20 simulation runs compared with theoretical distribution. The curves for the genuine units are on the right. The theoretical predictions for d_g are nearly hidden by the simulation data. Network parameters appear in Table 3.1, R = 1000.

match between simulation and prediction is excellent. The match for the low distribution is also good.

The d_g distribution can be approximated by a gaussian since it is binomial and Z in this case is not too far from 0.5; the compound binomial distribution for d_s can also be so approximated. Figure 3.4 shows the d_s frequency and a normal distribution with the same mean and variance – the match is quite good. However, in sparse networks the gaussian approximation would not be very accurate; as Z becomes small, the actual binomials should be employed to describe the distributions. Note that Palm (1981) inaccurately takes as his starting point the assumption that these distributions are gaussian[3]

Finally, notice that the frequency distributions for the low and genuine units overlap in Figures 3.2 and 3.3. Thus it is not possible to find a threshold based on dendritic sums such that all the genuine units fire and none of the low units do. Where to set the threshold becomes a design decision in the the trade off between false negatives and false positives.

3.3.2 Dendritic sum distributions using partial cues

The analysis of the dendritic sums for partial cues is a more general case of that for full cues. To express the distributions obtained using partial cues, substitute m_{cue} for M. For the low units

$$P(d_s = x) = \sum_{k=0}^{R} \binom{R}{k} \alpha_B^k (1-\alpha_B)^{R-k} \binom{m_{cue}}{x} (Z\rho(k))^x (1 - Z\rho(k))^{m_{cue}-x} \quad (3.2)$$

For the genuine units the dendritic sum is distributed $b(m_{cue}, Z)$.

The mean and variance of the d_g distribution are $m_{cue}Z$ and $m_{cue}Z(1-Z)$ respectively so they vary linearly with m_{cue}. The mean and variance of the d_s distribution also decrease monotonically as m_{cue} decreases, with the mean being approximately $m_{cue}Z\hat{\rho}$.

[3]Palm also states erroneously that the variance of the dendritic sum distribution for the low units is $m\hat{\rho}(1-\hat{\rho})$.

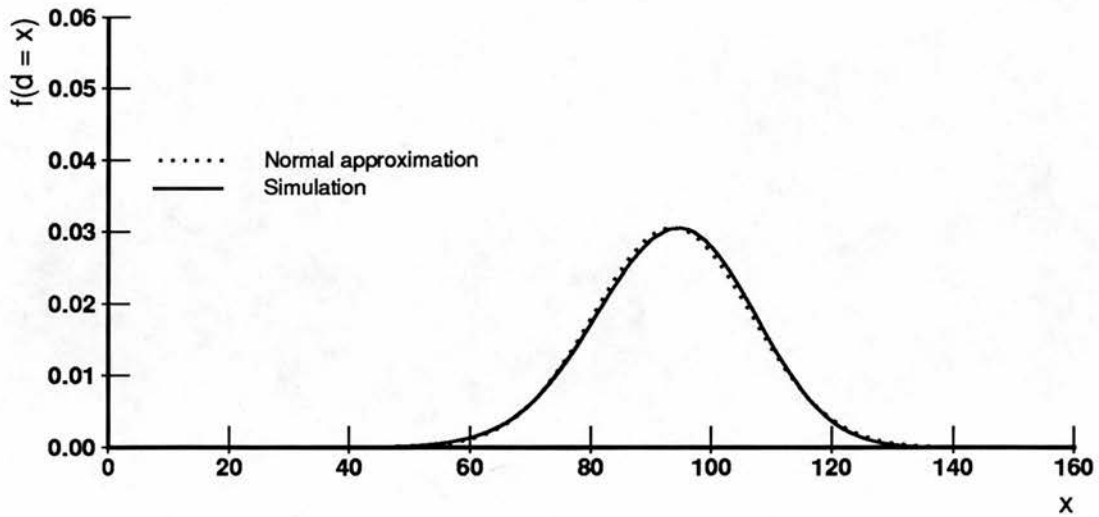

Figure 3.4: Dendritic sum frequency distribution for low units averaged over 20 simulation runs compared with normal distribution. Network parameters appear in Table 3.1, R = 1000.

3.3.3 Dendritic sum distributions using noisy cues

In the case of noisy cues, the dendritic sum distribution for the genuine units is different from the previous cases considered.

Let us define a noisy cue by the number of genuine and spurious bits in it: $m_{cue} = m_g + m_s$. Using a noisy cue for recall, the d_s distribution is exactly that described by equation 3.2, since for a low unit essentially all the active inputs impinging onto it are 'spurious'. For the genuine units the situation is more complex. The dendritic sum consists of contributions made by the genuine and spurious active input units. Each of the m_g active input lines from the stored input pattern that contacts an output unit which should be active will hit modified synapses (due to the Hebbian training algorithm). These make contact with that output unit with probability Z. The remaining m_s active input lines, the spurious bits, will hit modified synapses with probability $Z\rho_i$ for output unit i, just as in the case of the d_s distribution. The probability of d_g taking a particular value can be approximated by

$$P(d_g = x) = \sum_{x_g=0}^{x} \binom{m_g}{x_g} Z^{x_g}(1-Z)^{m_g-x_g} \times \left[\sum_{k=0}^{R} \binom{m_s}{x_s} (Z\rho(k))^{x_s}(1-Z\rho(k))^{m_s-x_s} \binom{R}{k} \alpha_B^k(1-\alpha_B)^{R-k} \right] \quad (3.3)$$

where x_g is the contribution to the dendritic sum made by the genuine input lines and x_s that made by the spurious; $x_s + x_g = x$.

Figure 3.5 shows the d_g frequency from a simulation run compared with the approximation in equation 3.3 (noisy cues were presented such that $m_g = 180$ and $m_s = 60$). The match is quite good. Note that this distribution is shifted to the left of that obtained using full cues (also shown in Figure 3.5).

This distribution can also be approximated by a normal distribution, as shown in Figure 3.6, where d_g frequency is plotted together with a normal distribution having the same mean and variance.

The mean of the contribution made by the genuine inputs is $m_g Z$ and an approximation to the mean contribution of the spurious inputs is $m_s Z\rho_i$. So a close approximation to the expected value of the dendritic sum of a genuine unit is $E(d_g) = m_g Z + m_s Z\rho_i$. The important point is that noisy cues elicit a much greater overlap between the d_s and d_g distributions than arises from using full cues. This makes recall less accurate.

3.4 Performance of thresholds setting strategies

Using the results of the last section, this section explores the performance of the partially connected associative net using thresholding strategies based on dendritic sums. The question addressed here is "How well can threshold strategies do in a single recall step in a heteroassociative net?"

To gain an initial view of the best possible recall performance using only dendritic sum information in the thresholding mechanism, an exhaustive search

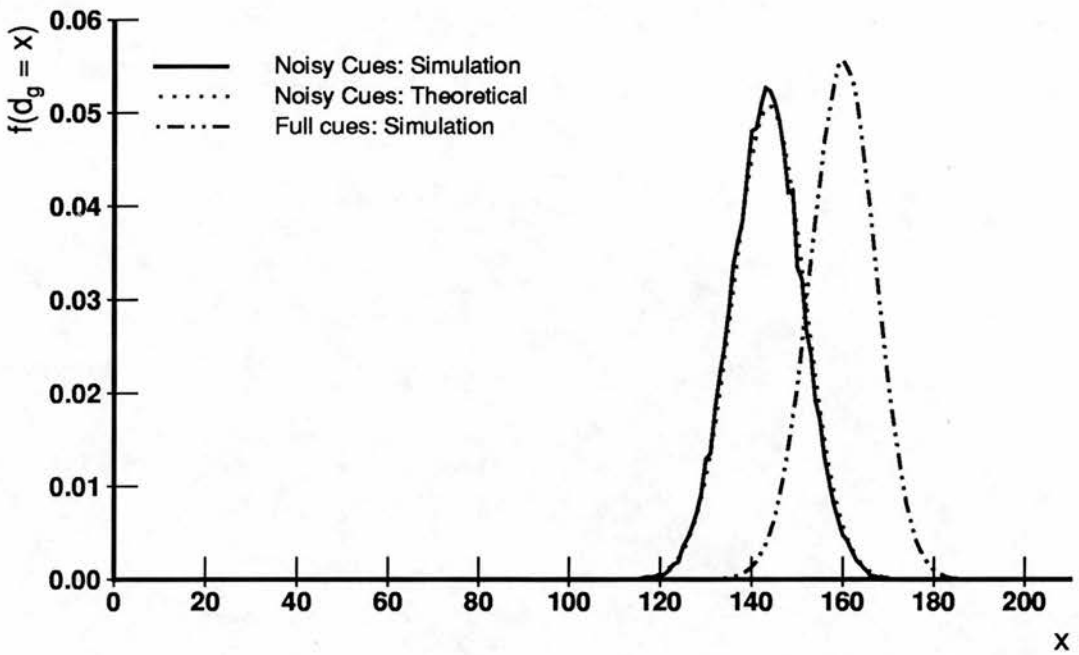

Figure 3.5: d_g frequency distribution with noisy cues compared with theoretical distribution. Network parameters appear in Table 3.1, $R = 1000$, and $m_g = 180$ and $m_s = 60$.

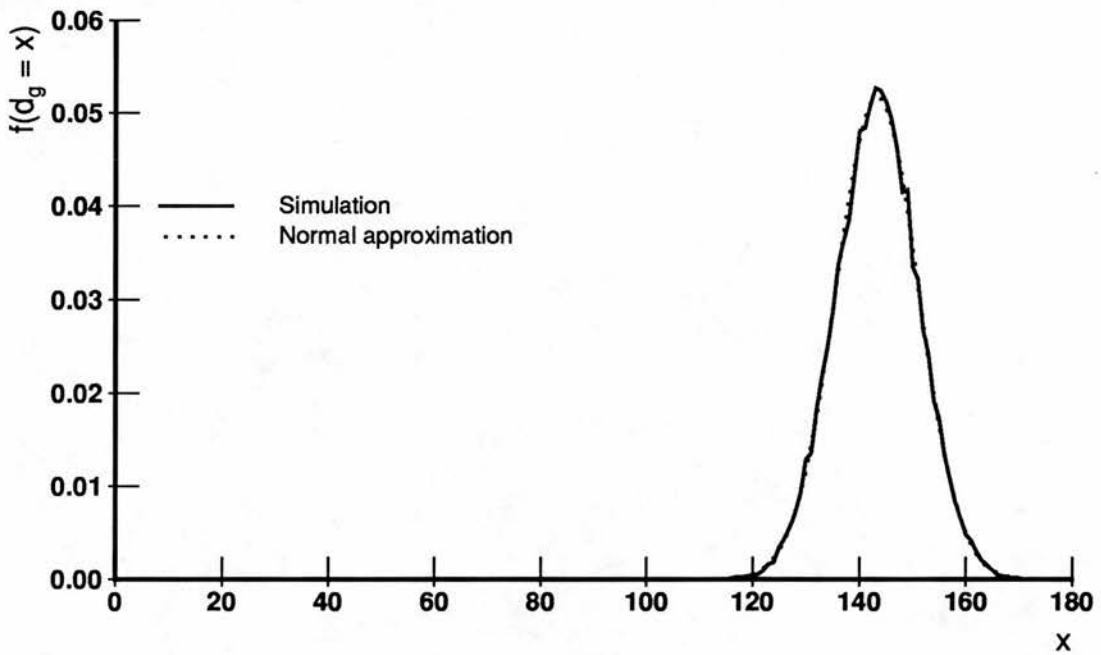

Figure 3.6: d_g frequency distribution with noisy cues compared with normal distribution. Network parameters appear in Table 3.1, R = 1000, and $m_g = 180$ and $m_s = 60$.

for the best threshold is employed. This strategy is *omniscient* in that in order to select the best threshold it has access to the target vector. Such strategies are useful in calculating the best possible performance, but are not applicable for practical applications of associative nets. Strategies which perform thresholding based on the dendritic sums can be developed which do not require knowledge of the target. In this thesis, they are called *practical* strategies to distinguish them from the omniscient ones. In this chapter two such strategies are discussed, one which exploits knowledge of ρ_i for an output unit and one which does not.

In the rest of this chapter, thresholding strategies will be developed and compared. For each of these strategies, simulations are performed using the example network described by the canonical parameter set shown in Table 3.1. In each simulation run, input and output pattern sets which contain patterns with the appropriate number of random bits on are selected, a number of pattern pairs is stored (usually 1000), then for each stored input pattern, cues with many combinations of missing and spurious bits are presented for recall. Recall is performed using the thresholding strategy being considered and data are stored about the number of genuine and spurious bits in the cue and output pattern elicited.

3.4.1 Omniscient exhaustive search

In order to gain an initial view of the recall performance possible using cues with varying numbers of missing and spurious bits, simulations were performed using the canonical parameter set of Table 3.1. 1000 pattern pairs were stored.

Under the assumption of omniscience, for each recall trial, a threshold T is chosen that will minimize the number of bits in error between the output and the desired target pattern (hamming distance). Thresholding is just a function of the dendritic sums, that is,

$$y_i = \begin{cases} 1 & \text{if } d_i \geq T \\ 0 & \text{otherwise} \end{cases}$$

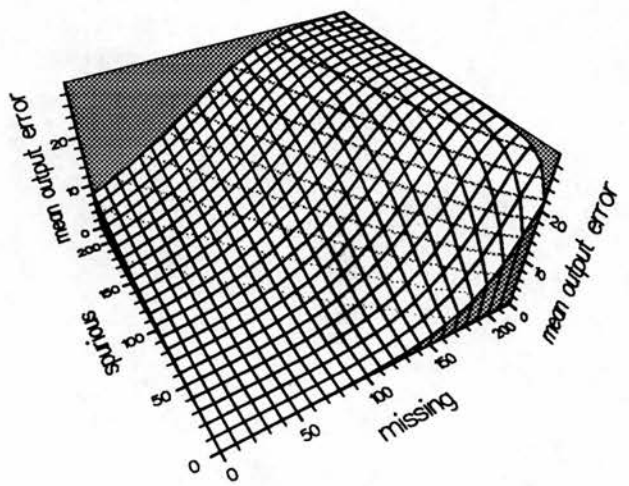

Figure 3.7: Recall errors as a function of missing and spurious bits in the cue. The number of missing bits is on the x-axis, spurious bits on the y-axis, and mean output errors on the z-axis. The 'Best T' strategy was used in the simulation. Network parameters appear in Table 3.1, R = 1000.

For an initial experiment, 1000 pattern pairs were stored in a net defined by the canonical parameter set. To test recall, partial and noisy versions of each stored input pattern were presented as cues. For each input pattern, cues were constructed with a number of genuine bits missing and a number of spurious bits added where the number of missing bits was $0, 10, \ldots, 220$ and the number of spurious bits was $0, 10, \ldots, 220$. That is, 23×23 cues were presented for each stored input pattern.

The results of these simulations are shown in Figure 3.7. In this 3 dimensional graph, the number of bits missing in the cue from the stored input pattern is represented on the x axis, the number of spurious bits added on the y axis, and the mean output error on the z axis. Mean output error is plotted for each combination of missing and spurious bits used to test recall. When both the number of missing and the number of spurious bits are low, the mean output

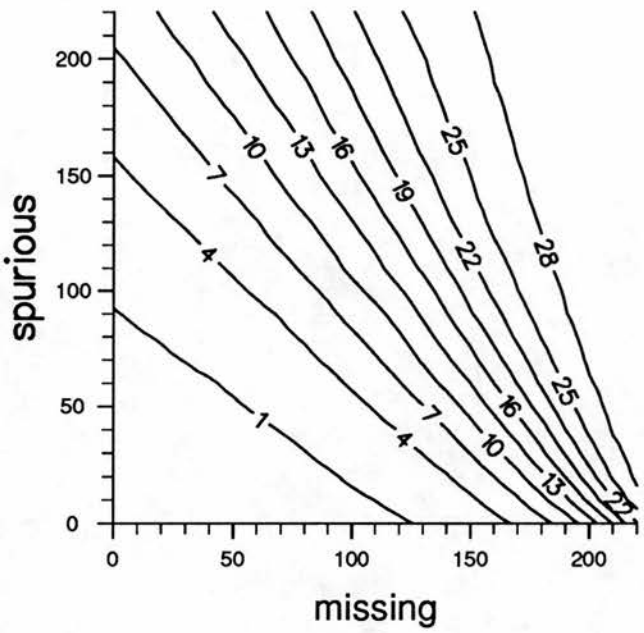

Figure 3.8: Iso-error recall cues. Curves are shown for cues which elicit equal output errors. Simulation parameters as in the previous figure.

error is also low and the surface it defines is relatively flat. As the number of missing and spurious bits increases, the output error increases steeply. To show which cues (defined by their numbers of missing and spurious bits) give rise to the same output errors, Figure 3.8 plots iso-error curves in the (δ_g, m_s) plane. For low output error, cues with similar hamming distance from their corresponding input pattern elicit similar output error, but large output error is only obtained when the number of missing bits in the cue is large.

It is not feasible to show simulation results for all of the cue configurations shown in Figure 3.7 in figures comparing the performance of several thresholding strategies so a subset of them are used. Cues that have equal numbers of missing and spurious bits will be used for the comparisons, that is, cues such that $m_{cue} = m_g + m_s$ with $\delta_g = m_s$. As seen in Figure 3.7, the results obtained using these cues provide a representative view of the recall performance of the net. They represent the range of cues from the ones that elicit low output errors to the ones that elicit high output error. In the graphs which compare the thresholding strategies, the hamming distance between the cue and the stored input pattern from which it came is represented on the horizontal axis, and the mean number of error bits in the output pattern on the vertical. Omniscient strategies are denoted by filled symbols and non-omniscient ones by unfilled symbols. As in previous graphs, solid lines denote simulation results and dotted ones theoretical values.

In Figure 3.9, The exhaustive search used above is labelled 'Best T'.

3.4.2 Practical threshold mechanisms using dendritic sums

The omniscient strategy discussed above provides a view of the performance that can be expected using thresholding based on the dendritic sums on the output units, but it is not useful in practical applications of associative nets. Two strategies are now developed that do not depend on knowing the target.

Threshold method based on d_s distribution

In section 3.3 it was shown that the dendritic sum distributions for the genuine and low units can be accurately described by expressions which are sums of products of binomials. The expression for the distribution for the low units is in terms of the number of active bits in the cue m_{cue}, the contact probability Z, the activity ratios for the input and target layers α_A and α_B, and the number of target patterns in which a unit fires, r_i. Each of these data items is available to a network. The expression for the distribution for the high units requires these and also the number of genuine and spurious bits in the cue. But it is not realistic to expect that these numbers will be available to the net. If m_g and m_s were known, the expressions for the d_s and d_g distributions could be exploited to set the threshold to minimize errors.

Consider the dendritic sum vector on the output units during a recall trial. What can the net do given that vector? Can it estimate the d_s and d_g distributions? For a single recall trial, since the number of genuine output units is small it is not possible to estimate the d_g distributions accurately – the data are simply too sparse. However, an estimate of the d_s distribution can be made. This idea forms the basis of the thresholding scheme now described.

The basic idea is to ignore the d_g distribution and simply try to minimize false positives. Given m_{cue}, Z, and $\hat{\rho}$, the mean of the d_s distribution can be estimated as $m_{cue}Z\hat{\rho}$. Then assuming the distribution is nearly normal and given the dendritic sum vector for a recall trial, the variance is estimated from that data. This estimate is used to set the threshold for a recall trial. (This system does not collect the dendritic sums of the low units across trials. The estimate is made using only the dendritic sum vector for a single trial.)

In the simulation shown in Figure 3.9, the threshold was set at $\mu_s + 3\sigma_s$ which gives a probability of spurious firing of approximately .0013. This strategy is labelled 'T:ds' in the figures. Not surprisingly, performance using this crude estimate is worse than that of the omniscient strategy, but it still works to an extent.

Threshold method based on dendritic sum distribution for the low units

The strategy described above estimates the dendritic sum distribution of the low units from the simulation data. The strategy presented here considers only the theoretical expression for the distribution when setting the threshold.

The dendritic sum for a low unit depends on its fraction of modified synapses, ρ_i, which in turn depends on the number of target patterns in which the unit fires. That is, units with different fractions of modified synapses should have different thresholds. We should be able to improve on the performance of the previous thresholding strategy if the threshold for a unit can depend on ρ_i.

The dendritic sum for a low unit which fires in k patterns is distributed $b(m_{cue}, Z\rho(k))$. The strategy is to set the threshold T_i for a unit so that

$$P(d_s \geq T_i) \leq P_{spur}$$

for some acceptably low value of P_{spur}. It is assumed that there is a lookup table for the theoretical d_s distribution.

Simulation results for $P_{spur} = .001$ are shown in Figure 3.9. The strategy is labelled as 'T:Pspur'.

3.4.3 Results of threshold strategies based on dendritic sums

Simulation results presented thus far in this thesis show that the partially connected associative net can function as a content-addressable memory using threshold strategies based on the dendritic sums on the output units. Even the strategy in which the parameters of the dendritic sum distribution of the low units are estimated from the dendritic sum vector even works, though not particularly well. Closer inspection of Figure 3.9 shows that once the hamming distance gets above 96 (48 missing and 48 spurious bits out of 240 active bits), the mean output error is well above 1. Using this strategy, the net functions as a content-addressable memory, provided that a substantial proportion of the content is there. The second strategy in which the threshold for a unit is set as a

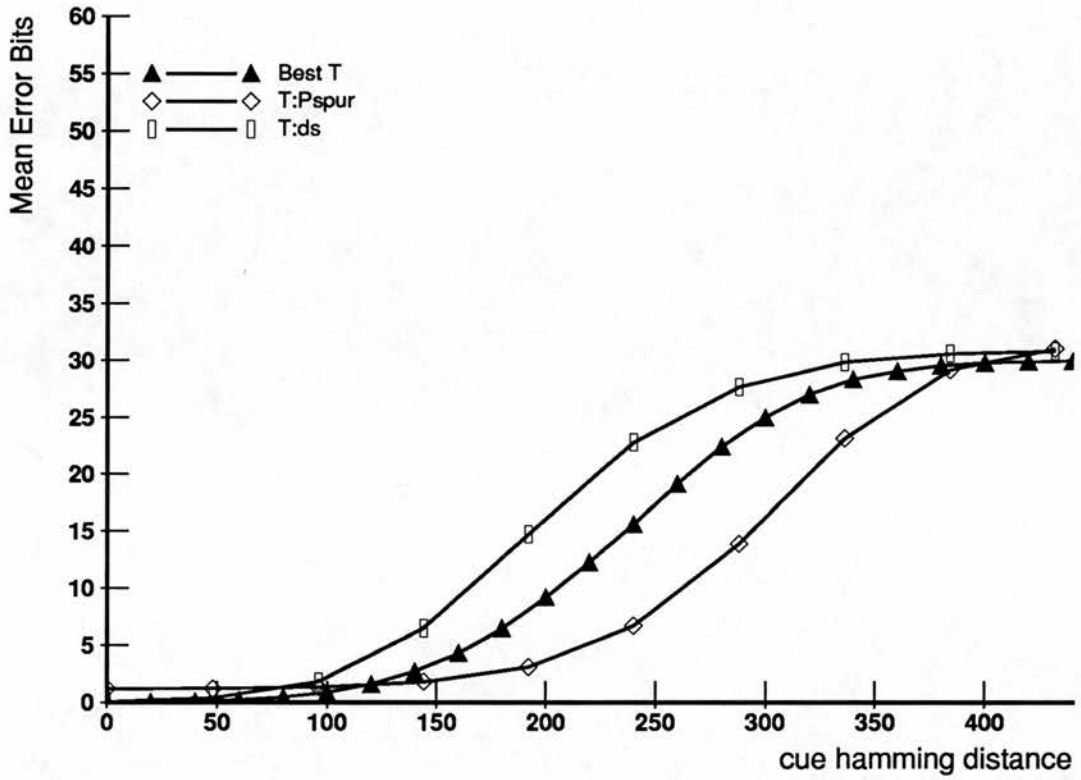

Figure 3.9: Comparing the T thresholding strategies. Recall errors using noisy cues plotted as a function of the hamming distance between the cue and the input pattern. Network parameters appear in Table 3.1, R = 1000.

function of the number of times it fires works much better than the first practical one described, and also outperforms the omniscient Best T strategy which did not exploit τ_i. Still, it only gives good recall for cues which are not very noisy; output errors are only low for cues with hamming distance less than 144. The most important point to note is that taking unit usage into account results in a dramatic improvement in performance.

The parameters of the example net were constructed with a view to storing 1000 pattern pairs with good content-addressable recall. Recall will be called good if the mean (or expected) hamming distance between output and target patterns is less than or equal to 1. Though somewhat arbitrary, it allows convenient comparisons with other associative memory work (Willshaw, 1971). In his theory of the archicortex, Marr (1971) stated that recall from his 'simple memory'

model, which is very similar to the partially connected net architecture here, should be possible using partial cues with ten percent of the genuine bits active[4]. Examination of Figure 3.7 shows that good recall can be obtained using some partial and noisy cues, but that partial cues with approximately ten percent of the genuine bits active yield high output errors. Hopefully thresholding strategies can be developed which improve on the performance of those based on dendritic sums.

3.5 Progressive Recall

So far this thesis has addressed recall performance in feedforward networks using one-step recall strategies (though the analysis developed can be applied to autoassociative networks). Gardner-Medwin (1976) investigated the characteristics of a recall strategy he termed *progressive recall* for partially connected autoassociative nets. The task to be performed is content-addressable recall using partial cues and the method involves reconstruction of a stored pattern in a number of steps. This also uses dendritic sum thresholds so is discussed here. In this section, the architecture of the network to which progressive recall applies is stated, then Gardner-Medwin's analysis is summarized. Simulations of several versions of progressive recall are performed and the results are compared with the expected values from analysis. The information efficiency of the net using progressive recall is calculated and discussed. Since information efficiency is a function of the amount of storage required by the net, it is also studied as a function of the connection density.

3.5.1 Overview of the algorithm and architecture

The network employed by Gardner-Medwin is very similar to the ones addressed thus far in this thesis. The only difference is that there is only one set of units. For recall, a partial cue is clamped onto the units as an input vector and a matrix multiply performed to yield a vector of dendritic sums. A thresholding

[4]Though he did not specify what good recall was.

operation is then performed. Basically, the threshold is set to allow more units to fire, but to keep the probability of spurious unit firing low.

3.5.2 Summary of the original analysis

Gardner-Medwin characterizes the behaviour of the progressive recall in the sparsely connected associative net via analysis. The key points of his analysis are summarized here, and then the actual performance of the recall process in simulations is shown in the next subsection.

First, the expected value for ρ is determined. As in the associative net

$$\hat{\rho} = E[\rho] = 1 - \left(1 - \frac{M^2}{N^2}\right)^R$$

The approximation

$$\hat{\rho} \simeq R\frac{M^2}{N^2}$$

is used which holds if $RM^2/N^2 \ll 1$. He notes that recall performance will degrade once $\hat{\rho}$ becomes greater than 0.5.

The distributions of the dendritic sums are assumed to be binomial:

$$P(d_g = x) = b(x; M, Z) \quad \text{for genuine units}$$

and

$$P(d_s = x) = b(x; M, Z\hat{\rho}) \quad \text{for low units}$$

These are modelled with Poisson distributions in the rest of his analysis. He considers nets with low Z and in this case the Poisson can be used to approximate the binomial.

He notes that where you set the threshold depends on what performance you are working to achieve, which is arbitrary to a large extent. In his analysis, he sets the acceptable probability of a spurious unit firing to to be quite low: $P_{spur} = 10^{-4}$. If the number of bits active in the cue is m_{cue}, the average dendritic

sum for a low unit is $m_{cue}Z\hat{\rho}$. The aim in setting the threshold is to find the minimum threshold T such that

$$\text{poisson}(x \geq T; m_{cue}Z\hat{\rho}) < P_{spur}$$

In progressive recall, the recall process proceeds over a number of time steps. For convenience, let us denote the step with a subscript on m (using t when a variable is needed). So let

$$\begin{aligned} m_0 &= m_{cue} \\ m_1 &= \text{The number of units active after the first recall step} \\ &\text{etc.} \end{aligned}$$

At each step choose the minimum integer T such that

$$\text{poisson}(x \geq T; m_tZ\hat{\rho}) < P_{spur} \tag{3.4}$$

If P_{spur} is low enough that the expected number of spurious units firing is less than 1, then the expected number of units active at step $t+1$ is

$$E[m_{t+1}] = m_0 + (M - m_0)\text{poisson}(x \geq T; m_tZ) \tag{3.5}$$

Gardner-Medwin calculates the theoretical performance of this procedure as a function of the loading $\hat{\rho}$. He asks how much of the stored pattern can be recalled given some initial partial cue. For a performance measure he uses

$$\frac{m}{M} - \frac{m_0}{M} = \frac{m - m_0}{M}$$

An important attribute discussed in the analysis is the expected value of the dendritic sum of the genuine units, $A = E[d_g] = MZ$. This value encapsulates both the activity level α and the density of connections Z since $MZ = M\frac{S}{N} = \alpha S$. MZ is also the expected number of active input lines impinging on any unit, which will be important in the treatment of the main model of this thesis. In Gardner-Medwin's paper (1976), the expected performance is calculated for A

= 5, 20, 50, and 200 for various network loadings. Performance is good when $A \geq 20$. Information efficiency is also calculated as a function of $\hat{\rho}$ for A = 5, 20, 50, and 200, and is best for A = 20 and 50. These results provide constraints on the architecture of the net for best performance. For good performance and information efficiency, MZ should be between 20 and 50. This finding provides an argument in favour of not only sparsely connected networks, but also sparse coding of the patterns.

3.5.3 Comments on the analysis

Let us consider this analysis in light of the work developed thus far in this thesis. Equation 3.5 predicts the expected number of units firing at step $t + 1$ faithfully if the dendritic sum distribution of the genuine units is binomial[5]. This means that the units that are active at step t must be genuine ones. If there were spurious units active, the distribution would be approximated by equation 3.3 instead of by a simple binomial. Thus low units must be kept from firing if equation 3.5 is to hold.

However, using equation 3.4 gives thresholds which result in a much higher probability of spurious units firing than P_{spur} since the dendritic sum distribution is not a simple binomial, but rather the sum of products of binomials given by equation 3.2. Using equation 3.2 to find thresholds that keep the probability of spurious units firing less than P_{spur} gives higher thresholds and hence reduces the expected number of genuine units recruited at a recall step. This may not be a problem – it may just require more steps to achieve good recall.

For autoassociative nets, the proportion of synapses modified during storage is slightly lower than in the heteroassociative case; effectively only $M - 1$ units are active in the 'input' pattern impinging onto an active unit. Thus

$$\hat{\rho} = E[\rho] = 1 - \left(1 - \frac{(M-1)M}{N^2}\right)^R$$

[5]Which can in some cases be modelled with a Poisson.

and

$$\rho(r) = 1 - \left(1 - \frac{M-1}{N}\right)^r$$

The dendritic sum for a low unit can be described by the binomial $b(m_t, Z\rho(r))$, where r is the number of stored patterns in which the unit is active. And so if it is possible to set the threshold for a unit as a function of unit usage, then this binomial can be used to find thresholds for individual units such that the probability of spurious unit firing is kept less than P_{spur}.

To make the discussion more concrete, let us consider an example. Let $N = 8000$, $M = 240$, and $S = 2000$. Let $P_{spur} = .0001$. Storing 600 patterns gives $\hat{\rho} = .416$. Using a binomial distribution to describe the d_s distribution[6], if a partial cue with 120 genuine bits active is presented to the net, the threshold T given by the original analysis is the lowest integer such that $b(x \geq T; 120, .25 \cdot .417) < P_{spur}$ which is 27. However, using this threshold in equation 3.2 gives $P(d_s \geq 27) = 0.000298$, which yields 2.3 expected spurious units after one recall cycle. To keep $P(d_s \geq T) < P_{spur}$, T should be set to 30. The difference between the thresholds selected is not large, but using thresholds that are too low over a number of recall cycles results in poor performance at the end of the process, as will be shown in the next section.

3.5.4 Comparing expected and simulation results

The dendritic sum distribution for the low units is described by equation 3.2 or $b(m_t, Z\rho(k))$ if unit usage information is available. We now address the question "What is the performance of progressive recall using knowledge of these distributions to set the thresholds?" via analysis and simulation. First, the performance at each cycle as recall progresses is plotted; expected values for the number of units firing after each cycle are calculated using the expressions for the dendritic sum distributions and are compared with simulation results. Next, recall performance as mean output error is plotted as a function of the network loading.

[6] At the high values of Z used in the examples in this thesis, the Poisson approximation does not hold so we use the binomials.

Progressive recall methods

There are a number of ways of interpreting the progressive recall concept. Four possible methods for performing it are:

m++: At each recall stage the threshold is set to just that value that will allow more units to fire than are currently firing; that is, find minimal T such that $m_{t+1} > m_t$. This method is denoted by m++ in the figures.

Pspur a: Use the original analysis and assume that the dendritic sum distribution for the low units is binomial:
Find minimal T such that $b(x \geq T; m_t, Z\hat{\rho}) < P_{spur}$.

Pspur b: Use the expression for the dendritic sum distribution for the low units (equation 3.2):
Find minimal T such that $P(d_s \geq T) < P_{spur}$.

Pspur c: Exploit unit usage and find threshold T for each unit such that $b(x \geq T; m_t, Z\rho(k)) < P_{spur}$.

The behaviour of each method is investigated using analysis and simulation. Simulation programs that implement each of these progressive recall methods were constructed and simulations were performed in order to investigate their performance in the content-addressable memory task.

The calculations and simulations were performed using parameter values $N = 8000$, $M = 240$ and $P_{spur} = .0001$. In most of the experiments, $Z = .25$.

Behaviour during progressive recall

This section considers the behaviour of progressive recall as the process continues over a number of cycles. For each recall cycle, the expected number of units active is calculated and the mean number of active units is taken from simulations. The case considered uses an initial cue with 48 genuine bits and no spurious bits and a net with $Z = .25$ after storing 400 patterns ($\hat{\rho} = .3$).

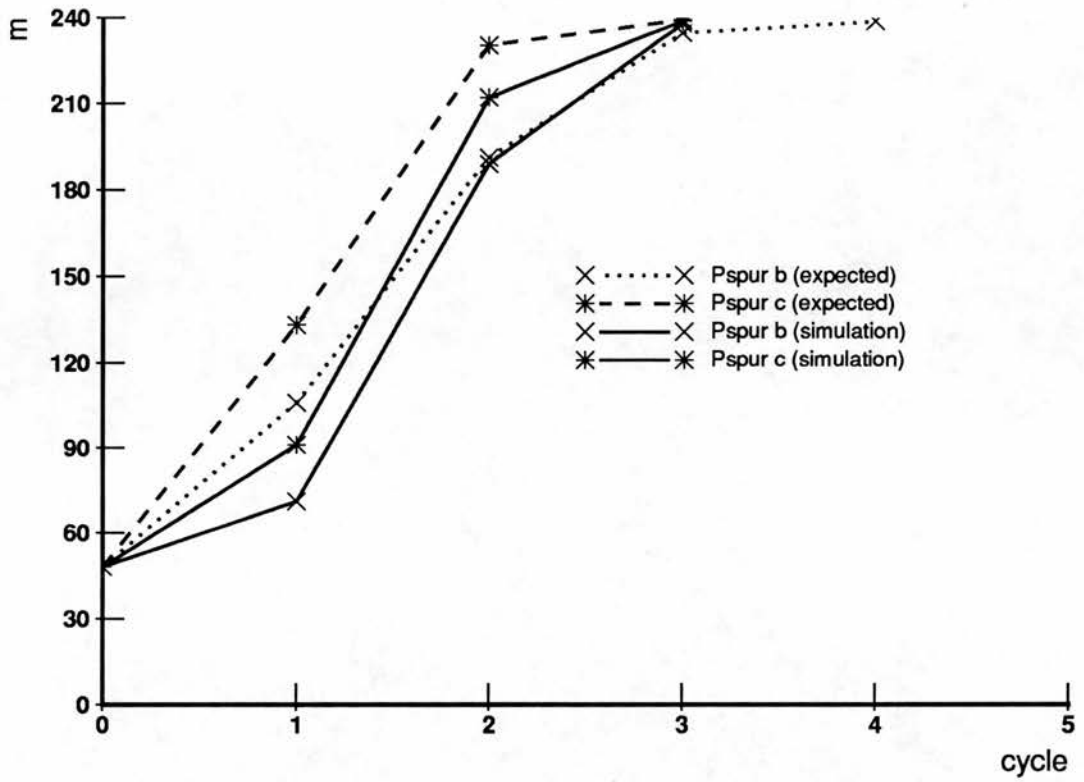

Figure 3.10: The expected number (calculated) and mean number (from simulations) of units active after each cycle of progressive recall. N = 8000, M = 240, Z = .25, R = 400, and $m_{cue} = 48$.

The expected number of genuine units active after each recall cycle is plotted in Figure 3.10 for methods 'Pspur b' and 'Pspur c'. The mean number of genuine units active after each simulation recall cycle is also plotted in the same figure for the same methods. Both the expected values and simulation results show that the number of genuine units active increases steadily with each recall cycle. At this loading, after only a few cycles the stored pattern is nearly recalled. In the simulations, the thresholds chosen do tend to keep spurious units from firing and in many trials recall is perfect, though the mean is 239 genuine units firing.

The 'm++' method results are similar, but since it is designed to allow the minimal number of units to be recruited at each cycle, many more cycles (approximately 20) are required so the results are not plotted in Figure 3.10 since

they do not fit into the horizontal scale.

Recall performance

This section shows how performance of the progressive recall methods considered here varies as a function of the network loading and the expected activity impinging on the units.

Using the 'Pspur b', 'Pspur c' and 'm++' progressive recall methods, the expected output error was calculated for A = 10, 20, 40, 60, 200 as a function of the number of patterns stored in the net for a initial partial cue with 48 active bits. A 48 bit cue was used because smaller cues do not elicit large enough dendritic sums to enable the methods to function well at all in the networks with small expected activity (low connection density). Figure 3.11 shows a graph for each of the methods. For all methods, the performance degrades as network loading increases. Performance is sensitive to the value of A; at most network loadings as A decreases below 40, performance degrades rapidly. When the expected dendritic sum for the genuine units $m_{cue}Z$ is low, the dendritic sum distributions for the genuine and low units overlap significantly which increases the expected number of errors no matter what thresholding strategy is used.

Simulations were run for all of the methods on a network with connection density Z = .25 (A = 60). In Figure 3.12 the top graph shows expected output error at A = 60 and the bottom graph the mean output error from simulations. The first thing to note is that the 'Pspur c' and 'm++' methods perform best. Next, the simulation results are consistently worse than the expected output error. The discrepancy arises from the assumption when calculating the expected output error that at each cycle of progressive recall the numbers of genuine and spurious units equal the expected values. Though the *mean* numbers of genuine and spurious units obtained in simulations are close to the expected values, the *actual* values come from distributions defined by the probabilities of false positive and false negatives, which can have large variances. A complete treatment of the dynamics of autoassociative networks is beyond the scope of this thesis so this discrepancy is not analyzed further.

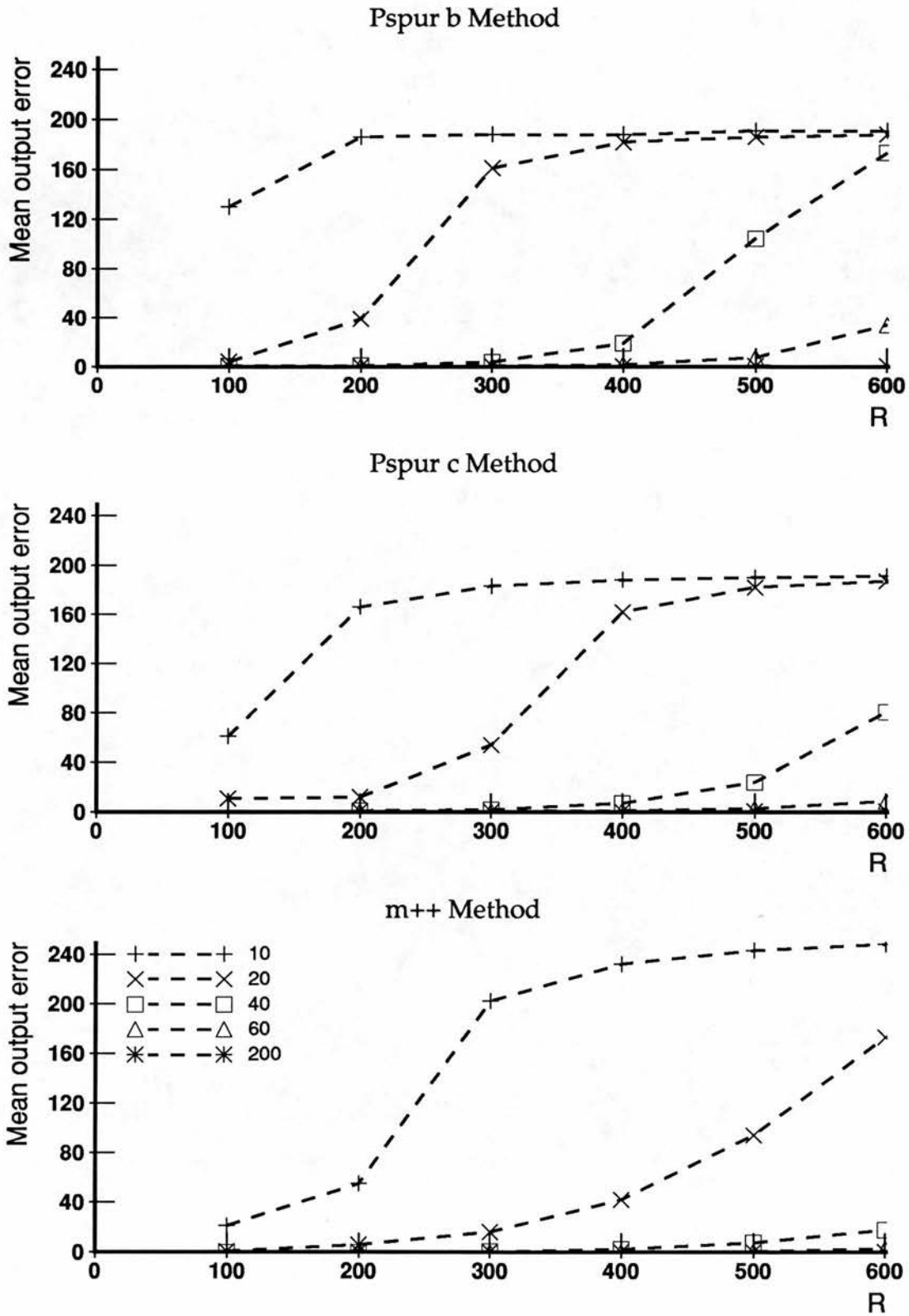

Figure 3.11: The expected output error using progressive recall as a function of the number of patterns stored. Graphs are shown for the 'Pspur b', 'Pspur c' and 'm++' methods. In each graph, values are plotted for A = 10, 20, 40, 60, 200.

Simulations were also performed for the method defined by the original analysis, 'Pspur a'. As noted above, the thresholds selected by this method are such that the probability of spurious unit firing is much greater than P_{spur}. Indeed, the output errors from simulations are high, certainly much higher than $(N - M)P_{spur}$.

Recall performance is relatively flat as a function of the initial cue size. This is shown in Figure 3.13 which plots mean output error as a function of the hamming distance between the initial cue and the stored pattern for several values of R. Results of the 'm++' method were used, which produces some of the best results. Poor performance using small initial cues is evident when using 'Pspur b', but not when using 'm++' at these network loadings. This flat performance as a function of cue hamming distance when using progressive recall is in marked constrast with that of the simple 'Best T' dendritic sum threshold.

3.5.5 Information efficiency of progressive recall

In this section, the information efficiency of the net using progressive recall is calculated. The amount of information required to identify the bits to be recalled given an initial partial cue is

$$I = \log_2 \binom{M - m_{cue}}{N - m_{cue}}$$

which is 1299.98 for the example when using a 48 bit recall cue.

The information efficiency is given by

$$\eta = \frac{R_{good} I}{NS}$$

where R_{good} is the number of patterns stored such that recall performance is acceptable. Let 'good recall' be recall such that the mean output error is less than or equal to 1. The information required to identify the 1 bit of error is not taken into account in this expression, but since it is small compared to I it does not change the efficiency much.

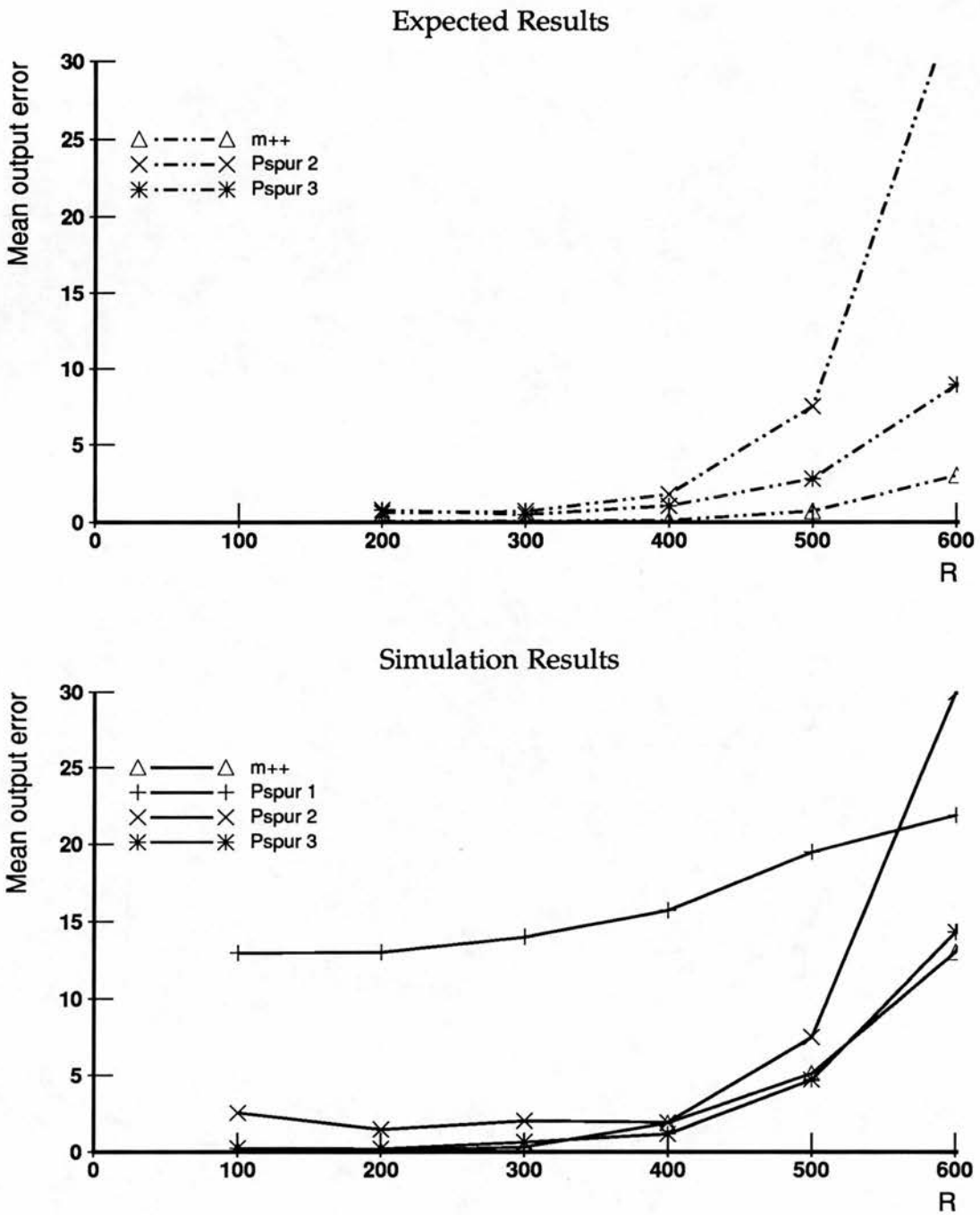

Figure 3.12: Comparing the expected and mean (simulation) output error. These graphs shows the output error using progressive recall methods 'Pspur b', 'Pspur c' and 'm++'. Expected output error is shown in the top graph and the mean output error obtained in simulations in the bottom one. For these graphs, A = 60 (S = 2000), N = 8000, M = 240, and $m_{cue} = 48$.

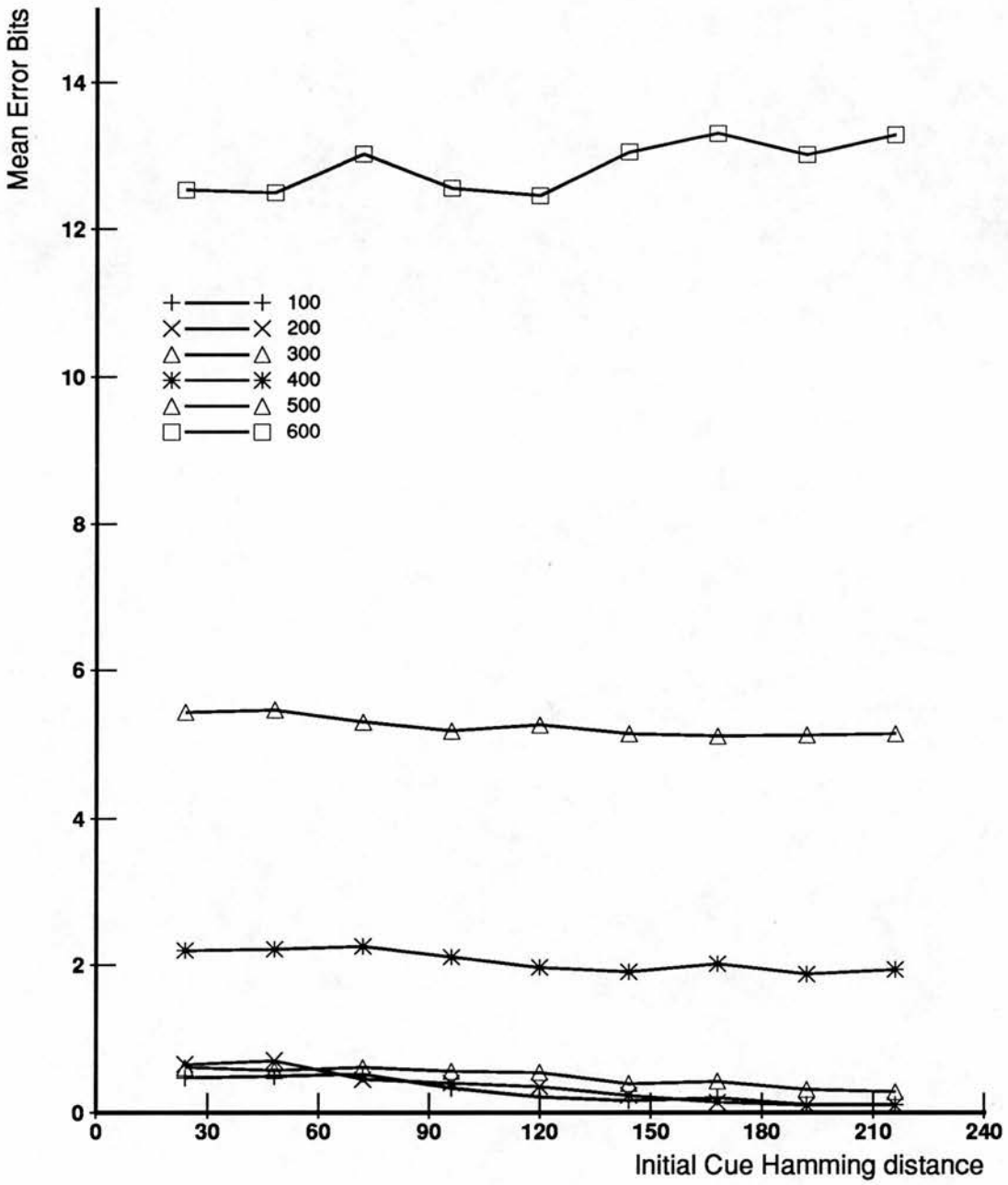

Figure 3.13: Mean number of error bits per output pattern as a function of cue hamming distance. Simulations were performed in which different numbers of patterns were stored; each curve in the graph corresponds to simulation results for one value of R = 100, 200, ..., 600. Simulation data are shown for the 'm++' version of the progressive recall algorithm. N = 8000, M = 240, Z = .25.

In the example net, simulation results showed that up to 300 patterns can be stored with good recall using the 'm++' method when $A = 60$. The information efficiency is then .024.

When considering partially connected nets, it is natural to ask how the connection density affects information efficiency. Increasing the connection density of a particular net increases the expected activity on the genuine units, A, and the net can store more patterns and still deliver good recall. But the increase in connection density means that more bits are required for the net. Let us study information efficiency as a function of the connection density.

Expected output error was calculated for the m++ method. As noted above, the expected output error is somewhat lower than the mean observed in simulations, but will serve to show how information efficiency changes as a function of connection density. The maximum number of patterns which can be stored such that the net still delivers good recall was calculated as function of $A = MZ$ and used to calculate information efficiency. This is plotted as a function of A in Figure 3.14. Since the effectiveness of thresholding based on dendritic sums depends on how much the dendritic sum distributions for the low and the high units overlap, what is critical is not M or Z alone, but MZ, since for a given network loading this gives an indication of the signal-to-noise ratio of the distributions. When $A = MZ$ is low, the distributions overlap more than when it is high. In these calculations, information efficiency is maximal at $A = 80$, where $\eta = .0414$. η drops off quickly as A goes below 40, but decreases slowly as A increases above 80. Unless A is below 40, changes in connection density do not affect information efficiency much.

3.5.6 Summary of progressive recall

We have seen that progressive recall can be an effective recall mechanism for a partially connected autoassociative net. It has the best performance of the dendritic sum thresholding strategies investigated in this chapter.

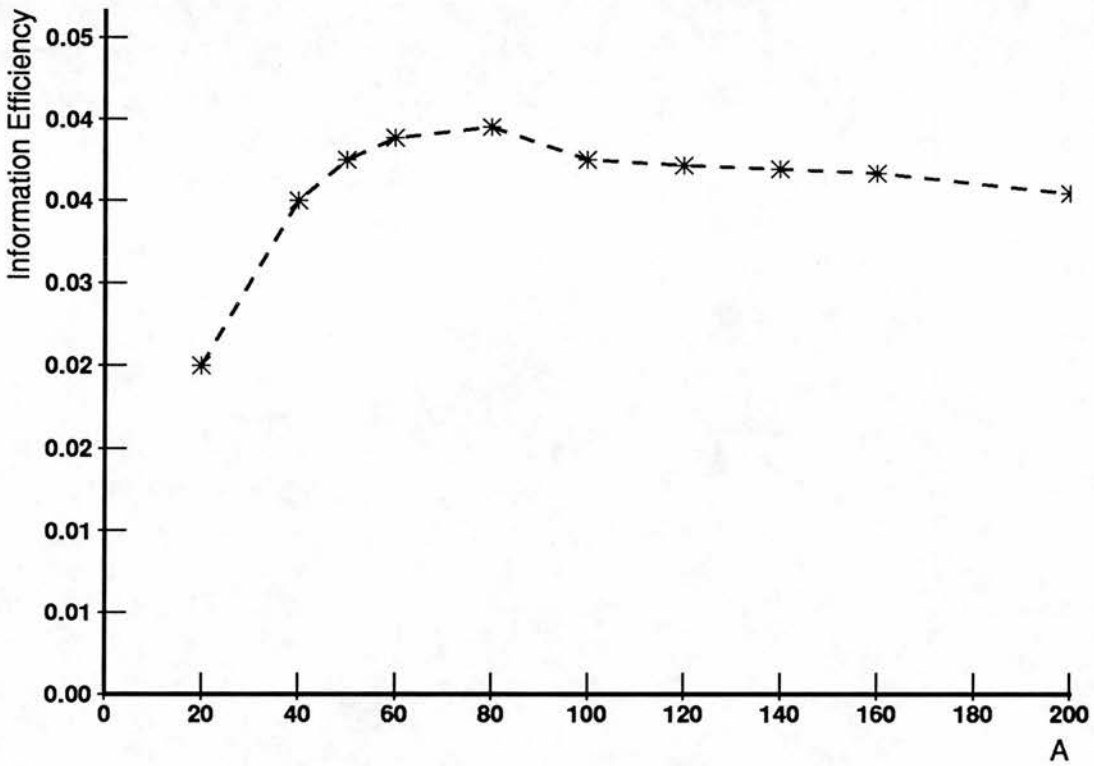

Figure 3.14: The information efficiency of a network using progressive recall. The expected maximum number of patterns which can be stored and still get good recall was calculated for the m++ method as a function of A and used to calculate information efficiency. For these calculations, N = 8000, M = 240, and $m_{cue} = 48$; S is varied to vary A = MS/N.

Chapter 4

Activity Based Thresholds

4.1 Introduction

In his theory of the archicortex, Marr (1971) posited that an estimate of the number of active input lines impinging on the dendrites of a primary unit will be available to it. Here the question is "How can activity information be exploited during recall in associative and content-addressable memory tasks?" Can thresholding schemes which make use of activity information perform better than the ones presented in Chapter 3?.

This chapter begins with an analysis of the dendritic sum distributions for a given activity value, then proceeds to develop and study thresholding strategies that make use of activity measures. It is found that the dendritic sum distributions for the low and high units are given by simple binomials so it is straightforward to calculate the probabilities of false positives and negatives for any thresholding strategy that has activity information available. Various thresholding strategies are suggested which exploit knowledge of activity in addition to other information available to the unit. They are divided into omniscient strategies and practical ones as in Chapter 3.

The rest of this chapter is devoted to the development and assessment of activity based thresholding strategies. We first explore Marr's suggestions regarding

thresholds that are functions of activity (Marr, 1971; Willshaw & Buckingham, 1990). To address the question of how well his ideas can work in principle omniscient strategies are developed. Turning to practical strategies, a strategy suggested by Marr is implemented in which the activity based threshold is slowly raised; this is called the 'Staircase strategy'. Gardner-Medwin suggests a thresholding scheme based on evidence theory. Two versions of that scheme are implemented – one which exploits r_i and one which does not. Finally we explore the performance of two new strategies based on the binomial distributions for d_s and d_g given knowledge of activity and unit usage.

4.2 Dendritic sum distributions for a given input activity

To understand what these thresholding strategies will be working with, let us examine the dendritic sum distributions of the output units when the activity impinging on them is known. Consider the case using noisy cues for recall such that $m_g + m_s = M_A$, where $m_s = sM_A$, $s \in [0, 1]$ and $m_g = gM_A$, $g = 1 - s$. At a fixed value of activity, a, the dendritic sum for a low unit involved in k patterns comes from the binomial distribution $b(a, \rho(k))$. In order to describe the distribution for a genuine unit, we need to consider the fraction of genuine and spurious bits in the cue. If a cue has no spurious bits, then each active line impinging on a genuine unit will contact modified synapses with probability 1, so the dendritic sum equals the activity. When $s > 0$, the spurious bits in the cue which hit the unit contact modified synapses with probability $\rho(k-1) < 1$ and so many of them will not contribute to the dendritic sum. Thus the dendritic sum of a genuine unit is likely to be less than the activity value when noisy cues are presented for recall. The probability of the dendritic sum equalling a particular value x is given by

$$
\begin{aligned}
&P(d_g = x \mid a) \\
&= \sum_{a_g=0}^{a} P(a_g) P(x_s = x - a_g) \\
&= \sum_{a_g=0}^{a} \binom{a}{a_g} g^{a_g} (1-g)^{a-a_g} \binom{a_s}{x-a_g} \rho(k-1)^{x-a_g} (1-\rho(k-1))^{a_s-(x-a_g)}
\end{aligned}
$$

For the genuine units, the distribution from simulations is slightly tighter than the theoretical one. In the simulations, the actual fraction of modified synapses on an output unit is on average slightly less than $\rho(k)$ (for an output unit active in k stored patterns); this is because the expression for $\rho(k)$ does not take into account the variability in the number of times an *input* line is active across the patterns stored. Thus the spurious bits in the cue which hit a genuine unit contact modified synapses with a probability slightly less than $\rho(k-1)$, which results in a distribution with a smaller variance than that of the theoretical one used here.

$$= \binom{a}{x}(1 - s(1 - \alpha_A)^{k-1})^x(s(1 - \alpha_A)^{k-1})^{a-x}$$

where a_g is the number of active input lines from genuine bits in the cue, a_s the number from spurious inputs, and $a_g + a_s = a$. Thus d_g are distributed $b(a, 1 - s(1 - \alpha_A)^{k-1})$.

The dendritic sum frequencies observed in simulations are in line with these theoretical statements. A simulation run was performed using a net defined by the canonical parameters (Table 3.1), storing 1000 pattern pairs. During recall, for each stored input pattern noisy cues were presented such that $s = .4$, that is $m_g = 144$ and $m_s = 96$. At each recall trial the frequency of each (a, d) pair for each unit is logged. The d_s and d_g simulation frequencies and theoretical distributions are shown in Figure 4.1 for $a = 160$ and $k = r_i = 30$; these are the expected values of a and r respectively and the data are representative of the match between simulation and theory. The simulation data and theoretical distribution are nearly indistinguishable for the low units, and the match is also good for the high units.

4.2.1 A pictorial view of the problem

This section presents a graphical view of the dendritic sum distributions as a function of activity. This provides an intuitive feel for how the low and high distributions might be separated. We have seen that statements about the dendritic sum distributions for the low and high units can be made given information about the number of active input lines and the unit usage. Several figures are now presented which provide an intuitive feel for how the low and high units can be distinguished given this information.

Simulation runs were performed in order to collect data on the dendritic sum (d) as a function of activity (a) and unit usage. The canonical parameter values were used (Table 3.1) and 1000 pattern pairs stored. Simulation runs were performed using noisy cues with $s = 0, .2, .4, .6$ ($\delta_g = m_s = 0, 48, 96, 144$, respectively). Figure 4.2 consists of one graph for each value of s used. Each graph is a

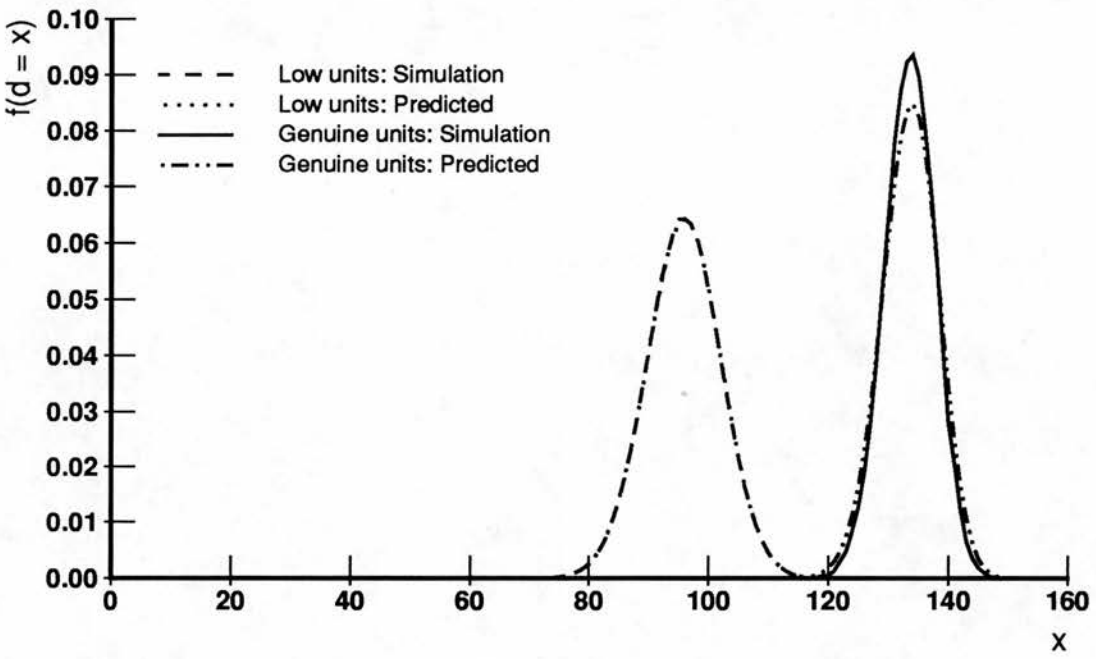

Figure 4.1: Dendritic sums for fixed activity: simulation frequency compared with theoretical distributions. Canonical parameters were used for the simulations and data are plotted for $m_g = 144$, $m_s = 96$, and $a = 160$, $R = 1000$, $r_i = 30$.

scattergram of the (a, d) pairs observed in the simulations[1]. For this graph, only data for units with $\tau_i = 30$ are plotted; the effect of varying τ_i is shown in Figure 4.3.

At s = 0, the scattergram consists of a line of points and a cloud of points below it. The line of points at $d = a$ is data from genuine units and the low units give rise to the cloud of points. As noted above, when there are no spurious bits in the recall cue, the dendritic sum of the genuine units equals the number of active input lines impinging on it. As s is increased, the cloud of points corresponding to the low units does not change, but the data for the genuine units now comes from the binomial distribution $b(a, 1 - s(1 - \alpha_A)^{\tau_i - 1})$, so the line of points observed at s = 0 drifts down and spreads out to form a second cloud.

With respect to thresholds, we can visualize how we would separate the genuine units from the low ones. At $s = 0$ thresholding is simple: units should fire if their dendritic sum equals their input activity, reminiscent of the fully connected associative net threshold rule. This is the first clear example of how knowledge of input activity can be exploited. That is, activity information makes thresholding simple for the case of cues with no noise ($s = 0$). Noisy cues present the more difficult, and more telling case. As s increases, it still appears that a straight line provides the best tool for separating the (a, d) data associated with the low and high units, though the data for the two sets of units overlaps more and more as s increases.

Figure 4.2 showed data for one value of τ_i. Figure 4.3 shows (a, d) data for several values of τ_i using noisy recall cues such that $s = .4$. The figure consists of several graphs - one for each of $\tau_i = 20$, 30, 40, and one for all output units combined. This view provides a feel for the usefulness of unit usage information to thresholding strategies; even visually it appears much more difficult to separate the clouds of points corresponding to all units than the clouds displaying data for a single value of τ_i.

[1]I would like to thank MR Frean, R Rohwer and AR Gardner-Medwin for helpful suggestions.

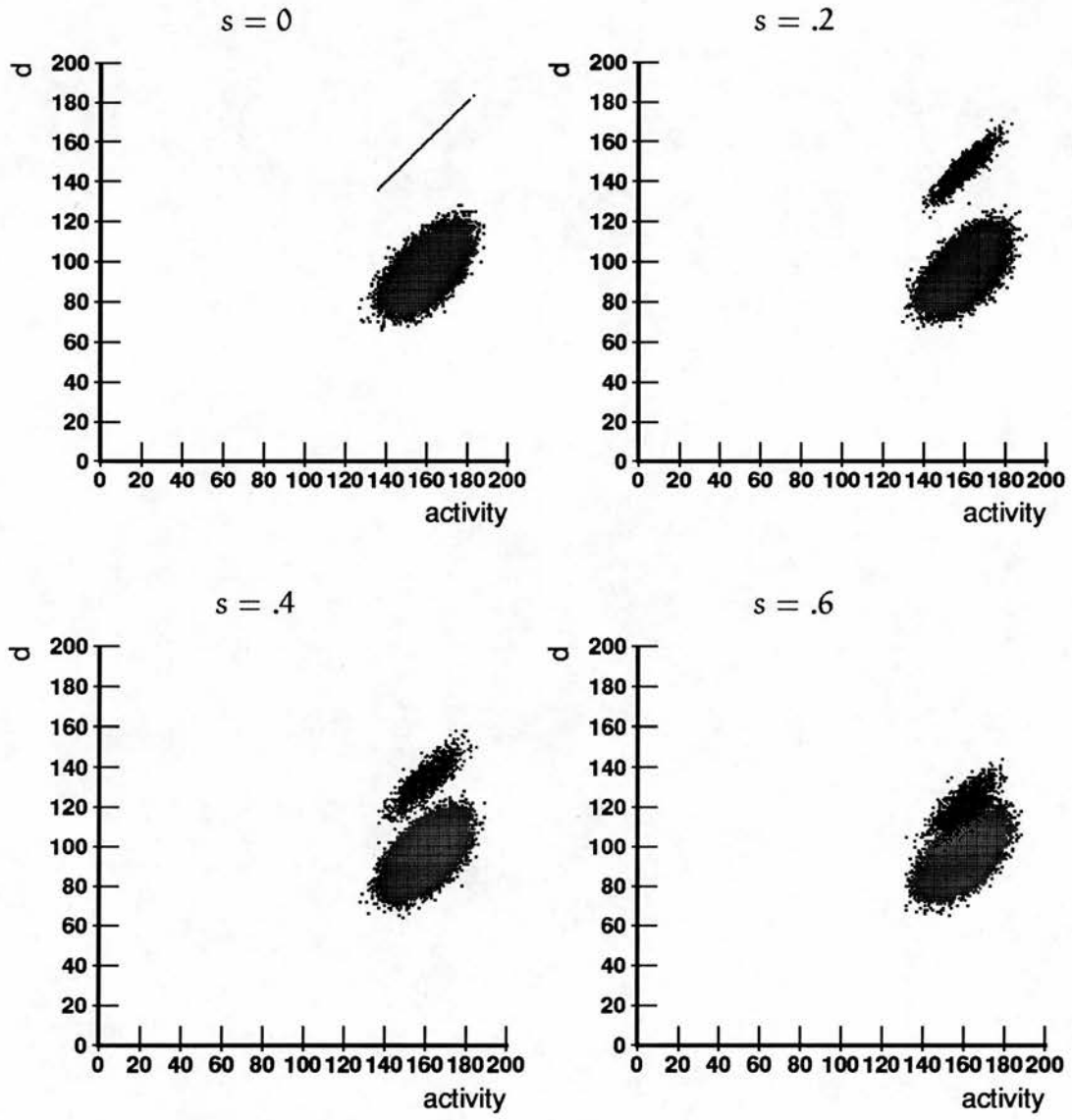

Figure 4.2: Scattergram of dendritic sums and activity observed in simulations. In these graphs, R = 1000, $\tau_i = 30$.

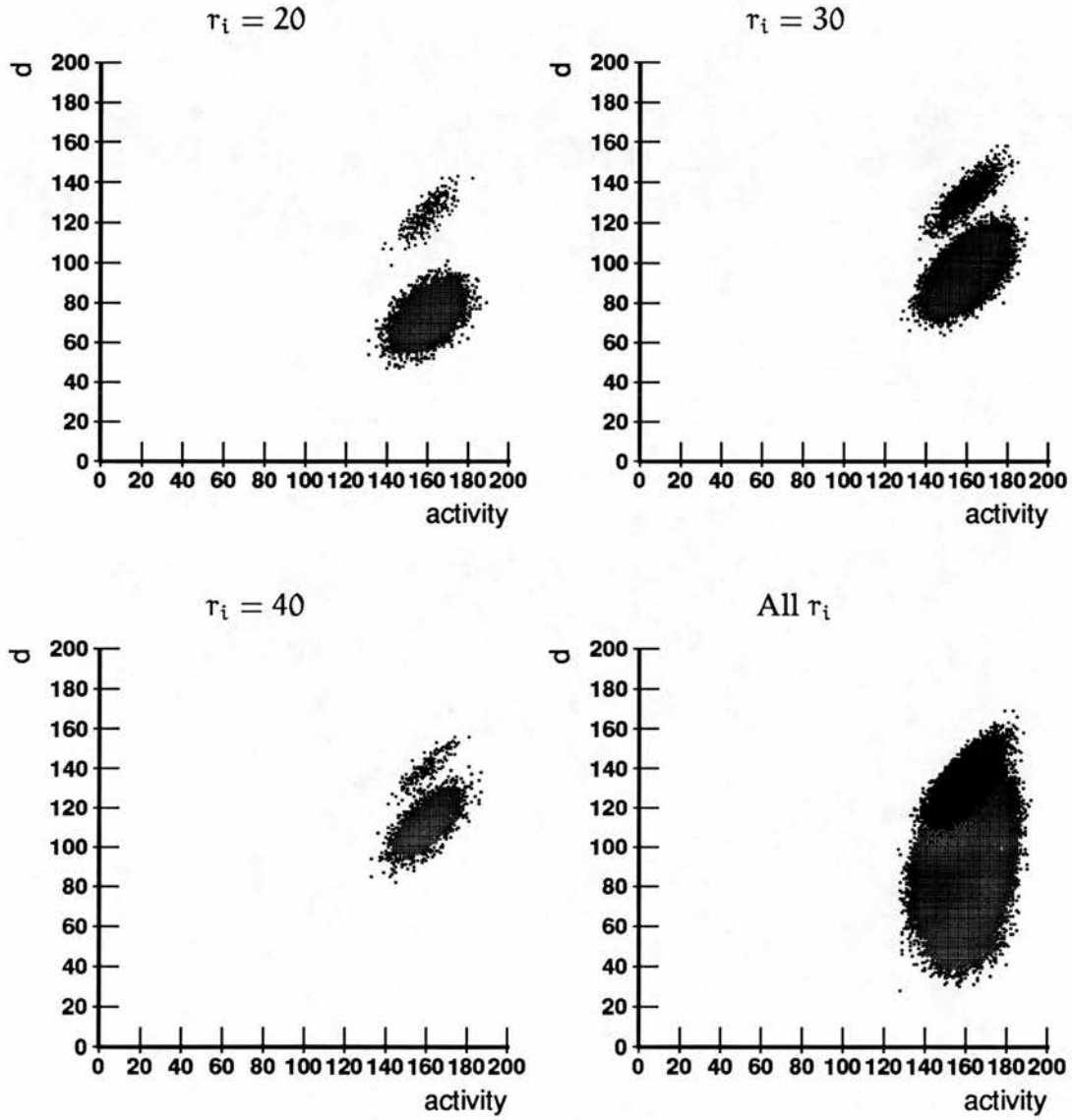

Figure 4.3: Scattergram of dendritic sums and activity observed in simulations for various values of τ_i. In these graphs, R = 1000, s = .4 (m_g = 144, and m_s = 96).

4.3 Thresholding strategies based on Marr's suggestions

In Marr's theory of the archicortex (1971), a primary unit fires during recall if a sufficient fraction (f) of its synapses receiving active inputs (input activity a) are modified and its dendritic sum exceeds a certain absolute threshold T. That is, the unit fires if $d_i \geq \max(T, fa)$. f is called the *division threshold* and T the *subtraction threshold*. To set these thresholds, Marr assumes that the cell can measure its dendritic sum and that another supporting cell provides a measure of the number of currently active inputs. An illustration of the way T and f thresholds can divide the low and high units is shown in Figure 4.4. The T threshold is representated as a horizontal line and the f threshold as a line passing through the origin with positive slope less than 1.

The question then is how to set T and f to minimize false positives and negatives. Marr notes that "Recovery of the whole of the simple representation depends on suitable juggling of T and f". Several thresholding strategies based on T,f thresholds are developed in this section, simulations using them are performed and the results are shown. The strategies to be examined are:

Omniscient strategies:

- An exhaustive search for best T,f – 'Best T,f'
- An exhaustive search for best T only – 'Best T'
- An exhaustive search for f only – 'Best f'
- An exhaustive search for f which exploits unit usage – 'Best f given r_i'

Practical strategies:

- Staircase
- f only

- T only

4.3.1 Omniscient strategies

As a first attempt to explore how well the dual T and f thresholds can perform, an omniscient exhaustive search strategy is developed. For each recall trial, the values of T and f are selected to minimize the number of output error bits.

Marr states that the combination of T and f thresholds will perform better than either on its own so simulations were performed using 'Best T' and 'Best f' strategies. The 'Best T' strategy is the one used in the initial simulations discussed separately in Chapter 3. In 'Best f', for each recall trial the f threshold which gives the the fewest output errors is chosen – the mechanism has access to the target vector in order to determine this.

Marr did not consider that the fraction of modified synapses would vary from unit to unit; he uses $\hat{\rho}$. The next thresholding strategy developed takes into account the number of target patterns in which an output unit fires and has a different f threshold for each. At each recall trial and for each value of r_i, the f threshold is selected that minimizes the number of false positives and negatives for that r_i. This is the 'Best f given r_i' strategy.

The results from simulations using each of these strategies are shown as filled markers in Figure 4.5. Nets defined by the canonical parameter values were constructed and 1000 pattern pairs were stored. Recall was tested using noisy cues with equal numbers of missing and spurious bits:
$\delta_g = m_s \in \{0, 24, 48, \ldots, 216\}$. The range of T and f tested is $T \in [1, m_{cue}]$ and $f \in \{0.05, .1, \ldots, 1.0\}$.

Not surprisingly, taking unit usage into consideration makes a great performance difference. The 'Best f given r' strategy performs much better than any of the others. The other three omniscient strategies: 'Best T,f', 'Best T' and 'Best f' perform similarly, with the omniscient 'Best T' performing somewhat worse than 'Best T,f' and 'Best f'.

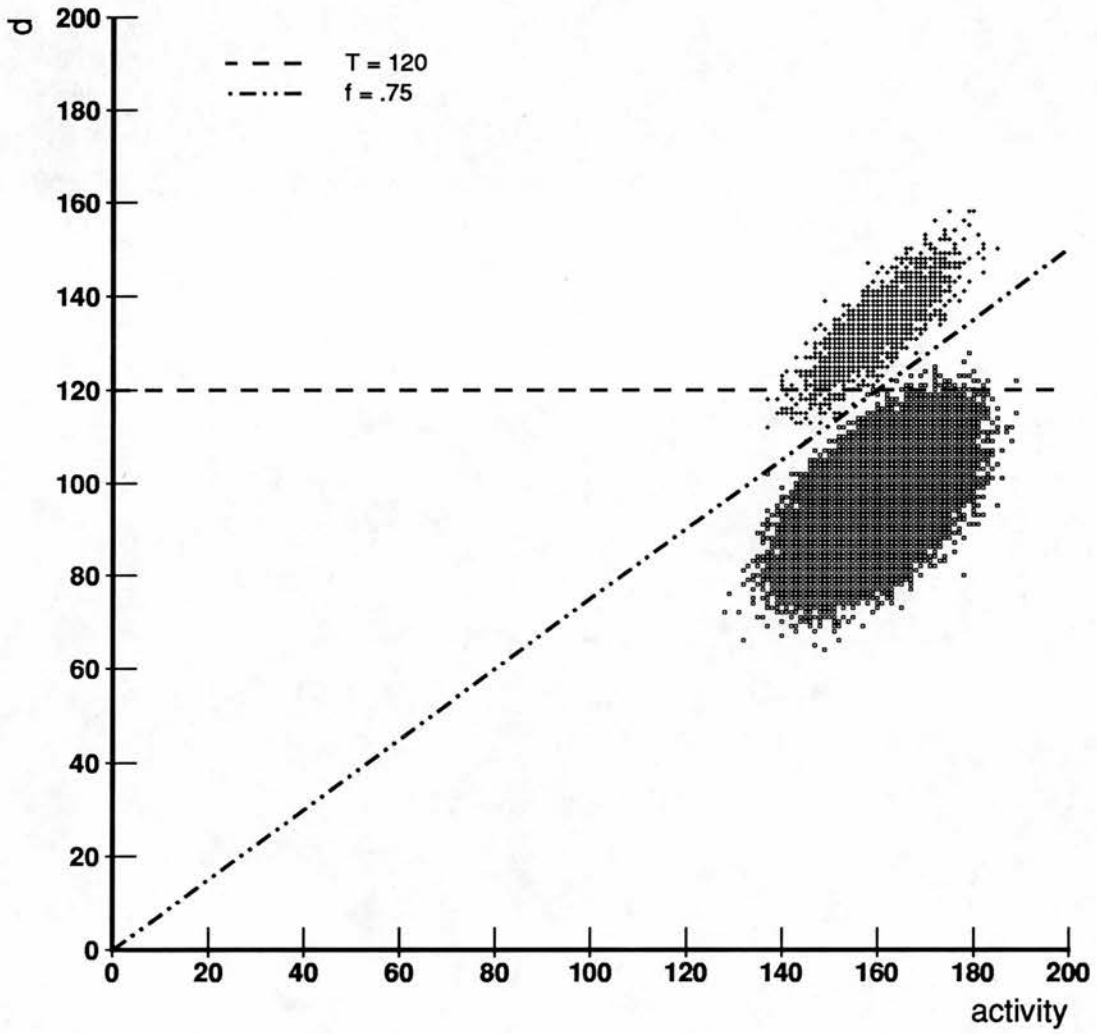

Figure 4.4: An illustration of T, f thresholds. The scattergram is of dendritic sums and activity observed in simulations. $m_g = 144$, $m_s = 96$ and $R = 1000$, $r_i = 30$, and $s = .4$.

4.3.2 Practical strategies

How can T and f be selected if the net does not already know the answer? Marr (1971) suggests:

> f must start low, and increase as the representation is recovered; T must decrease in such a way that the activity in $\mathcal{P}_3$ is kept roughly constant. (p 44.)

This is suggested in the context of the $\mathcal{P}_3$ to $\mathcal{P}_3$ collateral loop of his hippocampal model, but here it is implemented for a single recall trial and called the 'Staircase strategy'. Given the activity and dendritic sum vectors, for each f from 0.1 to 1.0, the algorithm finds a value T which gives the desired number of units firing. T starts high, which results in too few units firing, and is lowered until the desired number fire. As f is raised, T must be lowered to obtain M_B active units. For noisy cues, as f approaches 1.0 it is not possible to get M_B units firing. The highest value of f (and its associated T) that will still yield the desired number of active units is chosen. The order of varying f in this algorithm can be reversed and achieve the same functionality[2].

To examine the individual effects of T and f in the staircase strategy, simulations were performed using variants based on only T or f. In the case of the 'T only' strategy, for each recall trial, T was chosen to yield the desired number of active units (or as close to that number as possible). This corresponds to a *k-winner-take-all* strategy used in competitive learning (Rumelhart & Zipser, 1985). Analogously, for the 'f only' strategy, f was lowered until M_B units were firing (or as close to M_B as possible).

The results of these simulations are shown by unfilled markers in Figure 4.5. The practical strategies do not perform as well as the omniscient ones. Staircase and the *k-winner-take-all* 'T only' are approximately equal, with 'T only' performing relatively worse on the better cues. The performance of the 'f only' strategy is similar to that of the omniscient strategies on not-too-noisy cues and much

[2]This is what is done in the simulations.

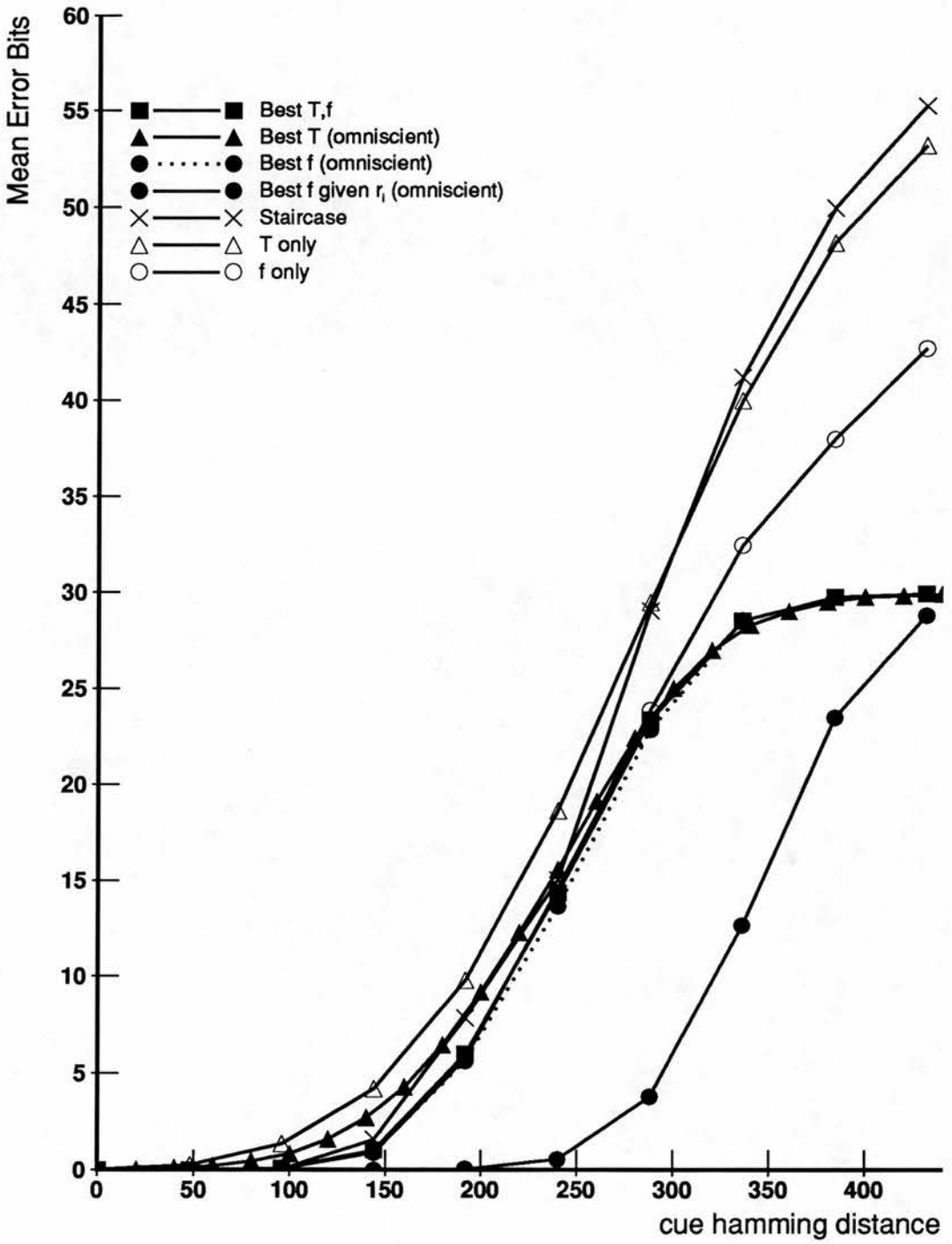

Figure 4.5: Comparing the performance of thresholding strategies motivated by Marr. Recall errors using noisy cues plotted as a function of the hamming distance between the cue and the input pattern. Filled markers denote the results from omniscient strategies and unfilled markers the results from practical ones. Nets defined by the canonical parameter values were used, R = 1000.

better than 'T only'. The fact that the f threshold outperforms the T threshold can be readily understood given the way the d_s and d_g distributions vary with activity α shown in Figure 4.2. An intuitive view of how to separate the clouds of points associated with the low and the high units suggests a line with slope near 1.0 and not a horizontal line. In this figure, an f threshold can be represented by a line with positive slope (≤ 1) passing through the origin while T thresholds correspond to horizontal lines.

The 'f only' strategy also outperforms Staircase. This result may seem a bit confusing at first: since the Staircase strategy exploits both T and f it would seem that it should be able to perform better than 'f only'. However, consider what all of these practical strategies are doing – they are working to minimize the difference between the number of desired units firing and the number that actually do fire, with respect to their different constraints. In 'f only', f is lowered until (nearly) M_B units are firing. In Staircase, f is also lowered, but at any particular value of f, T starts high and is lowered until M_B units are firing (or as many units as will fire at that f threshold). Examination of simulation logs reveals that in Staircase, a lower f is generally chosen than in 'f only' because the additional T threshold constraint in Staircase allows a lower f threshold to be chosen. The result is many more spurious units firing than in 'f only'. That is, the f threshold is doing most of the useful work in these simulations. However, later in this thesis it will be shown that a strategy very similar to Staircase works very well in progressive recall in an autoassociative net.

Referring to Figure 3.9, we see that 'f only' performs better than the practical strategies based on dendritic sums for most cues. For very noisy cues, the dendritic sum strategy which sets unit thresholds as a function of unit usage, 'T:Pspur', performs better than 'f only'. Again, this is because 'f only' sets thresholds to get the desired number of output units firing, while 'T:Pspur' sets unit thresholds to keep the probability of spurious unit firing less than 0.001. As noise increases, 'f only' allows more spurious units to fire in order to meet its goal.

4.4 Practical strategies based on evidence theory

Gardner-Medwin (1990) suggests that the weight of each synapse should come to reflect the probability that the unit should fire for a given level of activity impinging on it. He uses evidence theory to show that the firing threshold on the dendritic sum of a unit is a linear function of this activity. His argument is summarized here.

Consider a synapse on an output unit with an active input line impinging on it. The weight of that synapse should provide evidence that the unit should fire or remain quiet. If the unit is supposed to fire, the weight should be positive and if it is supposed to be off the weight should be negative.

For any given output unit in a layer with activity ratio α, if nothing else is known we can expect that it will fire a fraction α of the time. Gardner-Medwin defines the initial confidence that it should fire by

$$C_0 = \ln\left(\frac{\alpha}{1-\alpha}\right)$$

Let Condition X1 correspond to the case in which the unit is supposed to fire and Condition X0 correspond to the case that it should not. Let the weights take two states, H and L, corresponding to modified and unmodified. He defines H such that

> State H affords evidence *for* X1.

For a single *active* afferent he defines:

p_1	=	$P(H \mid X0)$	Probability of false positive errors, based on the state of just a single active afferent.
p_2	=	$1 - P(H \mid X1)$	Probability of false negative errors, based on the state of just a single active afferent.

For a single active afferent, the probability that it hits a modified synapse on a unit which should remain quiet, $P(H \mid X0)$, gives the probability that the unit

might falsely fire. Also, an active input hits a synapse in state H (modified) on a unit which should fire with probability $P(H \mid X1)$, so the probability that it hits an unmodified synapse on that unit is $1 - P(H \mid X1)$ which corresponds to the probability of that unit remaining falsely off with respect to a single active afferent. For good recall both p_1 and p_2 should be very low.

For each active afferent (Gardner-Medwin 1990):

$$\begin{aligned} P(H \mid X1) &= 1 - p_2 \qquad & P(L \mid X1) &= p_2 \\ P(H \mid X0) &= p_1 & P(L \mid X0) &= 1 - p_1 \end{aligned}$$

The weights of evidence for the unit firing given an active afferent striking a synapse in state H or L are given by:

$$\begin{aligned} W(X1 \mid H) &= \ln((1 - p_2)/p_1) = W_+ \\ W(X1 \mid L) &= \ln(p_2/(1 - p_1)) = -W_- \end{aligned}$$

The total evidence for condition X1 is

$$W(X1) = a_H W_+ - a_L W_-$$

where a_H and a_L are the number of active inputs that hit synapses with state H and L respectively, $a_H + a_L = a$. The confidence that the unit should fire is given by

$$C(X1) = C_0 + a_H W_+ - a_L W_-$$

Let us say that a unit should fire if its confidence $C(X1)$ is greater than some fixed value C. We then have

$$C_0 + a_H W_+ - a_L W_- > C$$

or rearranging terms

$$a_H > \frac{C - C_0}{W_+ + W_-} + a\frac{W_-}{W_+ + W_-}$$

In our model with binary synapses, state H corresponds to weight one and state L correspond to weight zero, so $d = a_H$. Thus the firing threshold on the dendritic sum is a linear function of the activity with slope $\frac{W_-}{W_+ + W_-}$ and intercept $\frac{C - C_0}{W_+ + W_-}$.

However, we have no means of determining what value of C is appropriate for minimizing false positives and negatives. Gardner-Medwin suggests that "varying a uniform inhibition over a population of cells ... (equivalent to varying $C - C_0$) allows any number of these cells to be selected".

4.4.1 Experiments with evidence theoretic threshold strategies

The performance of this application of evidence theory as a thresholding strategy was tested. It was interpreted in the most straightforward way: if the unit is supposed to fire then the genuine units in the cue which hit the unit hit synapses in state H with probability 1 and the spurious units which impinge on the unit hit synapses in state H with probability $\hat{\rho}$. So

$$P(H \mid X1) = 1 - p_2 = g + s\hat{\rho}$$

where s is the fraction of spurious bits in the cue and $g = 1 - s$. If the unit is supposed to remain off, each active input line hits modified synapses with probability $\hat{\rho}$ so

$$P(H \mid X0) = p_1 = \hat{\rho}$$

Putting these into the expressions for the W_+ and W_- gives

$$\begin{aligned} W_+ = \ln((1-p_2)/p_1) &= \ln\left(\frac{g + s\hat{\rho}}{\hat{\rho}}\right) \\ &= \ln\left(\frac{1 - s(1-\alpha_H)^f}{1-(1-\alpha_H)^f}\right) \\ W_- = \ln((1-p_1)/p_2) &= \ln\left(\frac{1-\hat{\rho}}{1-g+s\hat{\rho}}\right) \\ &= \ln\left(\frac{1}{s}\right) \end{aligned}$$

Thus W_+ is the log of the ratio of the means of d_g and d_s and is largest at $s = 0$ and W_- is zero at $s = 1$ and tends to infinity as s goes to zero.

Two strategies were implemented using these expressions.

Evidence theoretic: In this method, C_0, W_+ and W_- are calculated. Then given the activity and dendritic sum vectors the confidence that each unit should be active is calculated and placed in a confidence vector. A k-winner-take-all operation is performed on this vector to yield the desired number of units firing (M_B).

Evidence theoretic using τ_i: This version substitutes τ_i and ρ_i for $\hat{\tau}$ and $\hat{\rho}$ respectively in the expressions for W_+ and W_- when the confidence values are being computed ($\rho_i = \rho(\tau_i)$ is used).

Simulations were performed using these methods[3]. Results for $R = 1000$ are shown in Figure 4.6. As usual, the version that makes use of τ_i performs much better than the one that does not. Since the omniscient 'Best f given τ_i' strategy has the best performance so far, its simulation results are replotted in Figure 4.6 for comparison purposes. Performance of the 'Evidence theoretic using τ_i' is close to that of 'Best f given τ_i' until the cues become very noisy, when its performance declines.

This is the first practical strategy that works well. Its performance becomes worse for very noisy cues, but this is because in order to get the desired number of active units the confidence threshold for firing becomes dubiously low. Knowledge of the nature of the d_s and d_g distributions could be used in order to set a lower bound on C for a given probability of spurious firing which one is willing to accept, which could improve its performance. What information does this strategy require? It needs the activity (a_i), the dendritic sums (d_i), α_A, τ_i, and s. As noted above, it is not realistic to expect information about amount of noise in a cue to be available, which limits the practical usefulness of this algorithm.

[3]Using the canonical parameter values.

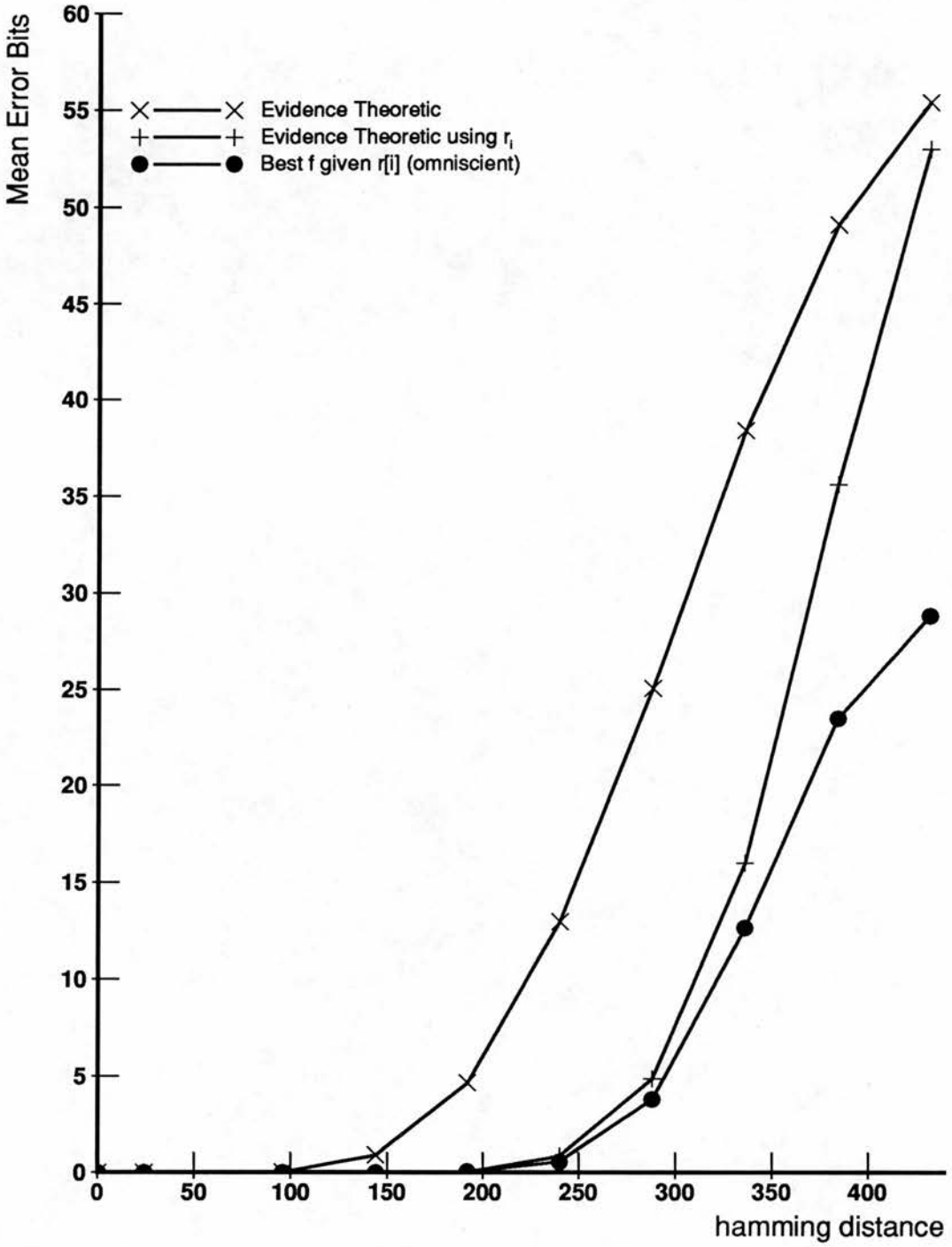

Figure 4.6: Performance of thresholding strategies based on evidence theory Recall errors using noisy cues plotted as a function of the hamming distance between the cue and the input pattern. Nets defined by the canonical parameter values were used, R = 1000.

4.5 Strategies based on the binomial distributions of the genuine and low units

We know that the dendritic sum distributions for the genuine and low units are simple binomials on activity for any particular value of ρ, but we have not yet exploited that knowledge. This section presents two new strategies which do. As noted above, we should not expect the net to have information regarding the fraction of spurious bits in the cue available to it. We want a thresholding strategy that works well across a wide range of noisy cues. The first strategy presented is analogous to the 'T:Pspur' strategy, but uses the distribution of the dendritic sums for the low units appropriate for the activity impinging on the unit. The second strategy exploits the distributions of both the low and the high units and works to minimize the expected number of false positives and negatives in the output. Both of them employ the expected dendritic sum distributions in order to calculate the probabilities of false negatives and false positives, but not the actual dendritic sum vector of the simulation.

4.5.1 A method based on the d_s distribution

The first strategy assumes that nothing is known about the dendritic sum distribution for the genuine units and simply tries to have the correct number of output units firing subject to the constraint that the thresholds are selected so that the probability of false positives does not exceed a given figure. Let P_{spur} be the probability of false positives that we will accept. The strategy basically starts with P_{spur} low and raises it until the desired number of units are firing. The method can be stated as:

- For $P_{spur} = 10^{-10}$ to 10^{-2}
  - Set the value of the number of active output units, m, to zero.
  - For each output unit i

* Lookup the lowest value of the threshold t such that $P(d_s \geq t) \leq P_{spur}$. That is[4], given a_i and ρ_i:
$\sum_{x=t}^{a} \binom{a}{x} \rho_i^x (1-\rho_i)^{a-x} \leq P_{spur}$.
* If $d_i \geq t$ then increment the number of active units, m by 1.

- Note the difference between m and M_B and keep track of the value of P_{spur} which minimizes this.
- If $m \geq M_B$, break out of the loop.

- At this point, the value of P_{spur} which results in the desired number of units firing (or as near to it as it could get) when units are thresholded using the t values associated with that P_{spur} has been determined.
- The output units are thresholded using the thresholds appropriate for that value of P_{spur}.

This method is labelled 'T:Pspur:a' in the figures.

4.5.2 A method based on both d_s and d_g: Guess s

If the fraction of spurious bits in the cue were known, this and the expressions for the dendritic sum distributions for the low and the high units could be used to calculate a threshold (t) for an output unit. This threshold could be set to minimize the expected numbers of false positive and false negative errors as a function of the unit's activity, fraction of modified synapses (or unit usage), and the fraction of spurious bits in the cue. In other words, t would be chosen to minimize

$$\begin{aligned} E[\Delta] &= (N_B - M_B)P(\text{false positive}) + M_B P(\text{false negative}) \\ &= (N_B - M_B)\sum_{x=t}^{a} \binom{a}{x} \rho_i^x (1-\rho_i)^{a-x} + \\ &\quad M_B \left(1 - \sum_{x=t}^{a} \binom{a}{x} (1 - s(1-\alpha_A)^{r_i-1})^x (s(1-\alpha_A)^{r_i-1})^{a-x}\right) \end{aligned}$$

[4]The binomials are not calculated explicitly each time.

Let us consider the thresholds that are chosen at a particular value of s. The thresholds selected when $s = .4$ are plotted as a function of activity together with the clouds of (a, d) points from simulations in Figure 4.7. We see that these thresholds separate the clouds well. The thresholds which are selected for several values of s are shown in Figure 4.8 as a function of activity. Across most of the range of activity these thresholds define straight lines[5] but as activity decreases the thresholds curve downwards towards the $y = x$ line. Figure 4.9 shows the threshold as a function of s for $a = 160$ and $r_i = 30$. As s increases from 0, the threshold decreases to minimize the expected number of false positives and negatives as the distribution for the genuine units shifts down. Then as s approaches 1, the threshold increases. By this point the dendritic sum distribution for the genuine units is so close to that of the low units that the best thing to do is raise the threshold to lower the expected number of false positives in the output. The slope of the curve defined by the thresholds in Figure 4.9 is greatest for low and high values of s and relatively flat for intermediate values of s.

We may not know how noisy the cues presented to the net are, but we do know that when $s = 0$ the dendritic sum d for a genuine unit equals its activity a. As s increases, the expected value of d decreases for the genuine units. If the threshold '$d = a$' is used and the cue is noisy, it is likely that fewer than M_B units will fire. The thresholding strategy developed here exploits this.

The basic idea is to guess the fraction of noise bits in the cue, and then checks how many units would then fire with thresholds set appropriately for that noise level. If the s guess is too low, it is likely that less than M_B units will meet the thresholds. This strategy starts by using $s = 0$, and slowly increases s until the desired number of units fire[6]. So far the goal is to get the correct number of units firing, but it is also desirable to minimize the number of false negatives. As the s guess increases, the probability of false positives (P_{spur}) tends to increase. In the process of calculating the thresholds, the probabilities of false positives are also calculated for each relevant value of activity and unit usage. It is useful to choose a value of s such that the average P_{spur} is low.

[5]Which makes implementation of them significantly less compute intensive than it might be otherwise.

[6]Or as close to M_B as possible.

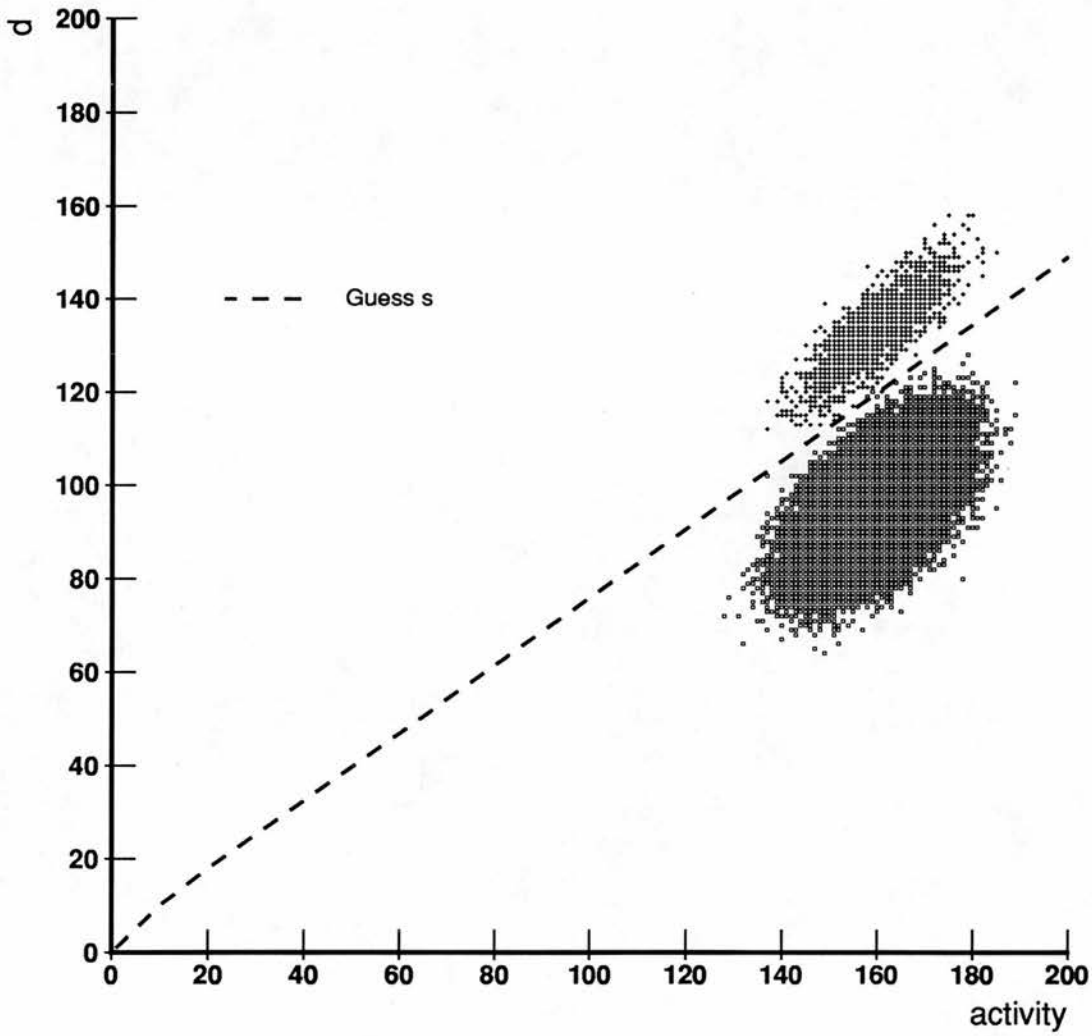

Figure 4.7: Guess s thresholds together with (a, d) data from simulations. The scattergram is of dendritic sums and activity observed in simulations. R = 1000, $r_i = 30$, and $s = .4$ ($m_g = 144$, $m_s = 96$).

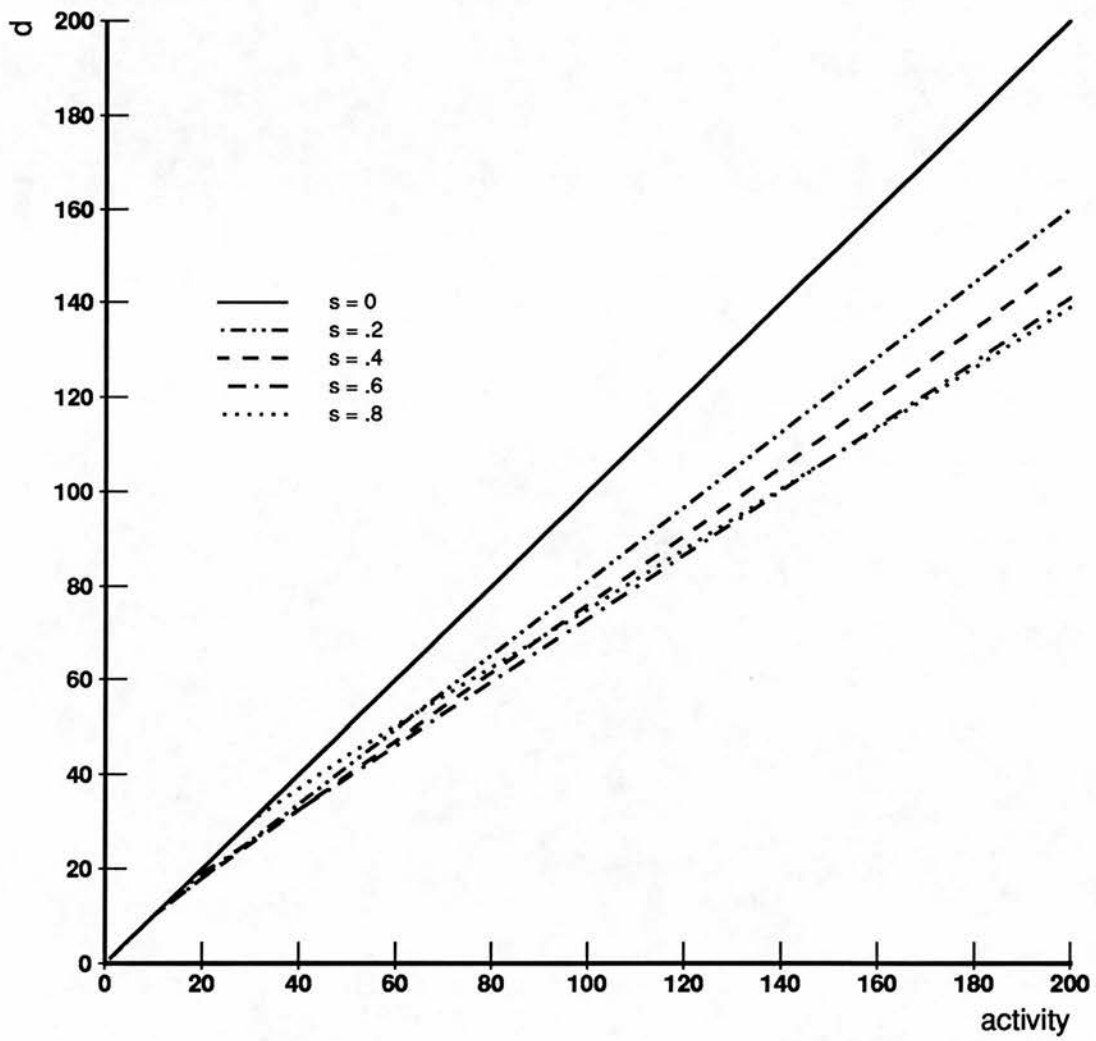

Figure 4.8: Thresholds chosen by the Guess s strategy as a function of activity for several values of the fraction s of spurious bits in the cue. One curve is plotted for thresholds for each value of $s = 0, .2, \ldots, .8$. The thresholds for $r_i = 30$ are shown.

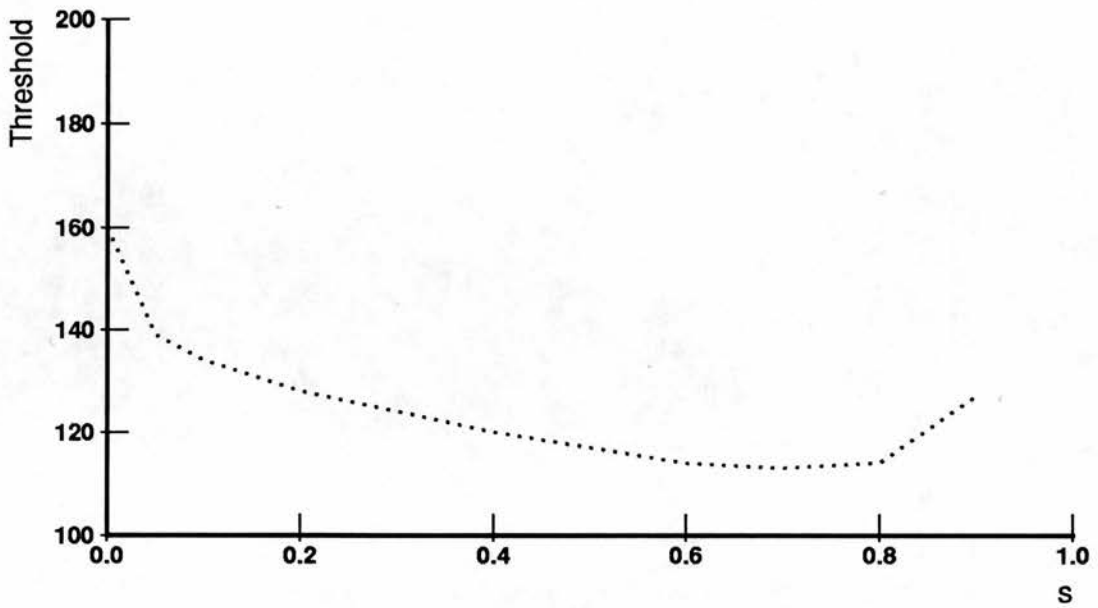

Figure 4.9: Thresholds chosen by the Guess s strategy as a function of the fraction s of spurious bits in the cue for units with $a = 160$, $\tau_i = 30$.

A high-level statement of the method is:

- For $s = 0, \ldots, .95$
  - Set the value of the number of active output units, m, to zero.
  - For each output unit i
    - Lookup the threshold value t appropriate for a_i, τ_i, and s that minimizes the *expected* number of false positives and negatives.
    - If $d_i \geq t$ then increment m by 1.
  - Note the difference between m and M_B and keep track of the value of s which minimizes this.
  - Note the average probability of false positives, $\langle P_{spur} \rangle$, at these thresholds.
  - If $m \geq M_B$ or $\langle P_{spur} \rangle > 0.01$, break out of the loop.
- At this point, the value of s has been determined which results in the desired number of units firing (or as near to it as it could get) when units

are thresholded using the t values calculated for that value of s.

- These thresholds are applied to the output units.

4.5.3 Results of the strategies based on the binomial distributions of the dendritic sums

Simulations were performed using 'T:Pspur:a' and 'Guess s'. Networks were constructed using the canonical parameter values and 1000 pattern pairs were stored. Results of recall trials using noisy patterns are shown in Figure 4.10. Both of these strategies perform better than any of the other practical strategies. The only strategy that consistently outperforms these two is the omniscient 'Best f given r_i'. For very noisy patterns, the 'T:Pspur:a' strategy performs worse than Guess s. Inspection of the simulation results show that Guess s is consistently better with respect to minimizing expected errors, which is what it is designed to do. For very noisy patterns, the average P_{spur} of the units for the value of s which gives the best value of m in the 'Guess s' strategy is consistently lower than the P_{spur} value chosen by the 'T:Pspur:a' strategy. In both, for very noisy patterns ($s > .7$) the number of active output units is much less than that desired. This is reflected in the high output error – most of which is due to false negatives.

Though these strategies are similar, they differ in several ways which affect recall performance. In the 'T:Pspur:a' strategy, P_{spur} is constant across all values of activity of the units, while in the Guess s strategy, s is held constant while testing the guess, but P_{spur} varies inversely with the activity of a unit. These strategies do not work well when activity is low – the distributions for the low and high units are too close together and far too coarse grained for effective threshold setting.

Since the Guess s strategy is based on the binomial distributions which describe the dendritic sums (for particular values of activity and unit usage), it should be possible to make some analytic statements about its recall performance in terms of mean error bits. If we assume that the noise level s which Guess s arrives at corresponds fairly closely to the actual noise level in the cue (which turns out

to be the case), the expected value of the output error can be calculated across the range of values for activity and unit usage. Using the canonical parameter set and $R = 1000$, simulation results and the expected output error are plotted in Figure 4.11. The match between theory and simulation is excellent across most of the range of noise level. The expected error is somewhat less than the simulation mean output error for very noisy cues. The situation is just the one desired: there exists a thresholding strategy for partially connected associative nets that yields good performance in the content-addressable memory task and whose performance is amenable to analysis.

The results are good enough to encourage further exploration of the performance of these networks using this recall strategy. First, to broaden the view of recall performance for different cues, Figure 4.12 shows mean output errors in the canonical net for cues with $\delta_g, m_s \in \{0, 24, 48, \ldots, 216\}$ from a simulation in which 1000 pattern pairs were stored. The recall errors are low across a wide range of missing and spurious bits in the cue. Recall is almost perfect for partial cues. For cues with many missing bits, errors increase steeply as spurious bits are added to the cue. Comparing this figure with Figure 3.7 makes clear just how much better Guess s is than the omniscient 'Best T' that does not exploit unit usage or activity information.

4.5.4 Discussion of all thresholding strategies

Chapter 3 and this chapter have presented a number of thresholding strategies for partially connected associative nets. These strategies have different motivations, and the recall performance of the net varies significantly across them. The amount of information required also varies between strategies. The performance of a set of the strategies representative of the different motivations is replotted in Figure 4.13.

The strategies were presented according to their motivation – those based primarily on the dendritic sum vector first, then those motivated by Marr's theory of the archicortex, then those which exploit knowledge of input activity. They could also be ordered with respect to how much information they require. Most

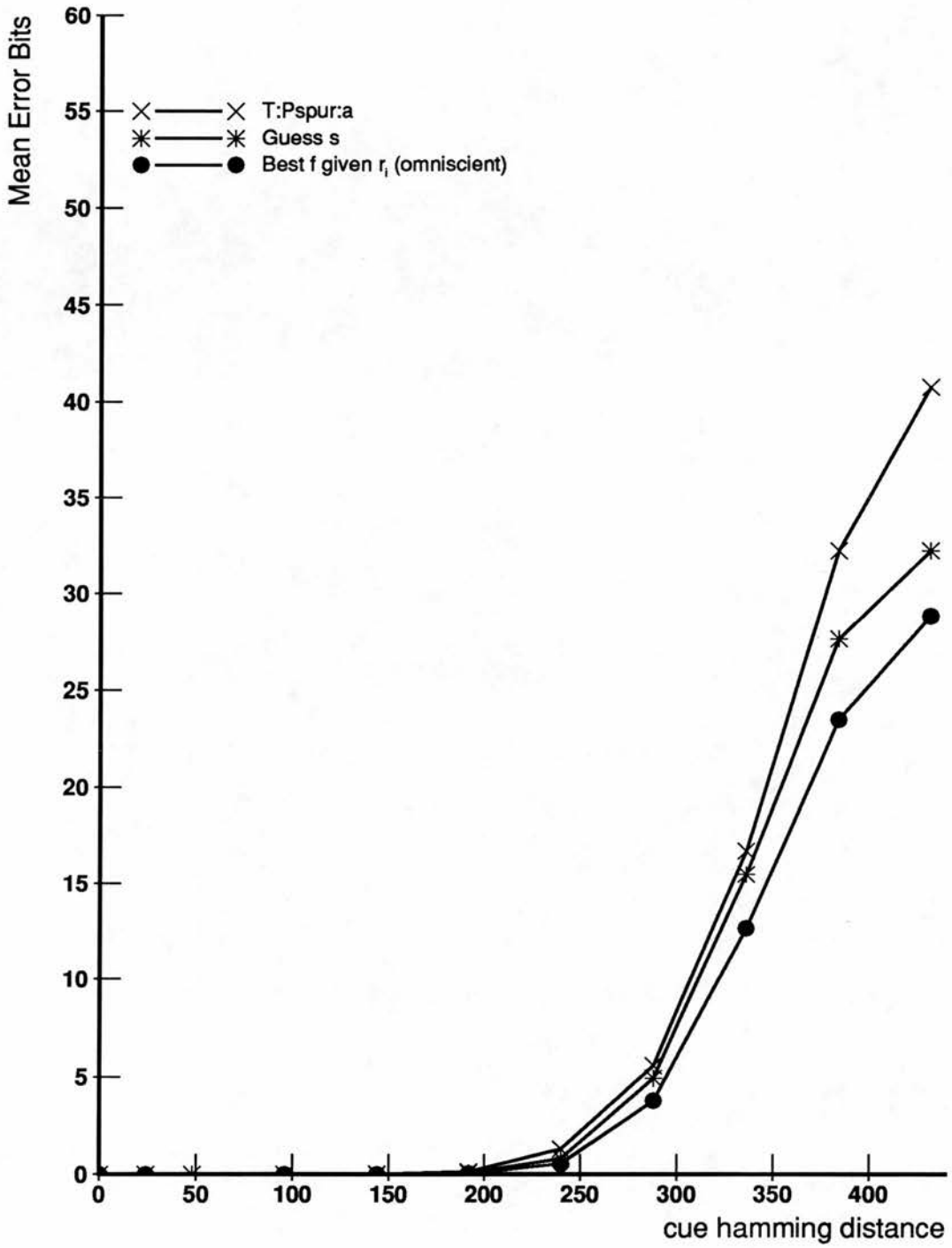

Figure 4.10: Performance of the thresholding strategies based on the binomial distributions which describe the dendritic sums of the low and genuine units. Recall errors using noisy cues plotted as a function of the hamming distance between the cue and the input pattern. Nets defined by the canonical parameter values were used, R = 1000.

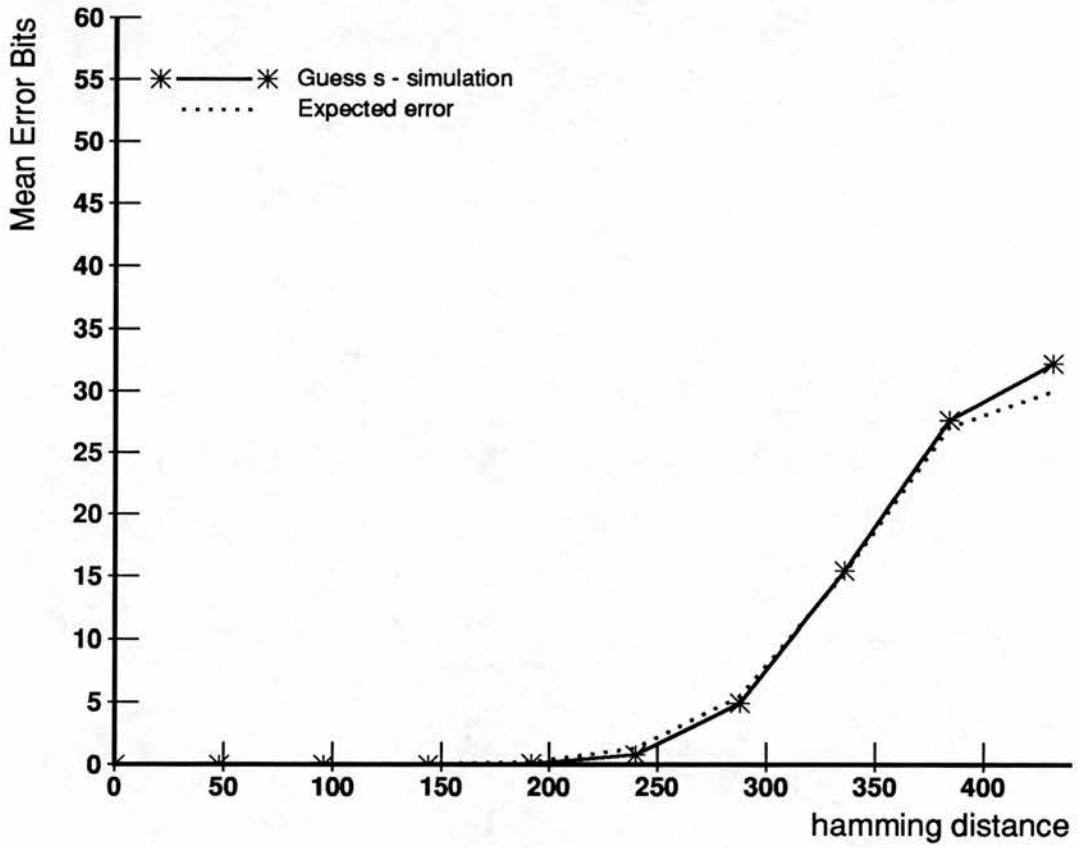

Figure 4.11: The Guess s strategy: Comparing simulation results with theoretical expectation. Recall errors using noisy cues plotted as a function of the hamming distance between the cue and the input pattern. Nets defined by the canonical parameter values were used, R = 1000.

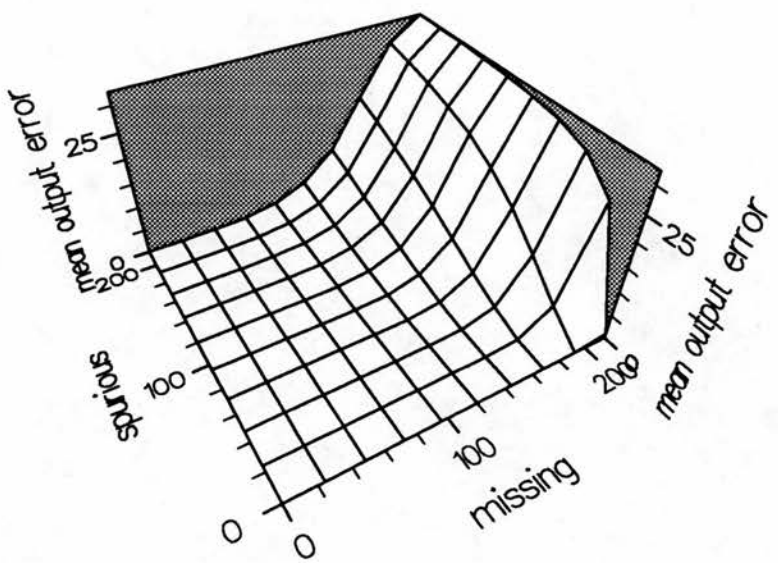

Figure 4.12: Mean output errors as a function of missing and spurious bits in the cue using the Guess s strategy. The number of missing bits is represented on the x axis, spurious bits on the y axis, and mean output error on the z axis. Nets defined by the canonical parameter values were used, R = 1000.

of them require the knowledge of number of output units which should fire – the goal of almost all of the strategies is to minimize the difference between the actual and desired number of units firing, subject to other constraints. Thus the network must have some way of measuring the number of units which would fire at any threshold settings. The simplest strategy is probably 'T only'. It makes use of only the dendritic sum vector, the number of desired active units, and the number which do fire at a particular threshold setting. The recall performance of the net when using this strategy is not very good. The first practical strategy presented in Chapter 3, 'T:ds', which estimates the low unit dendritic sum distribution based on the dendritic sum vector, requires more information. Built into it is the assumption that this distribution is normal (gaussian) and knowledge of the characteristics of a gaussian, in particular, that the area under the density curve to the right of the mean plus three standard deviations is very small. Its performance is similar to that of many other strategies for very noisy cues, but is quite poor for better cues.

The 'T:spur' is in many ways the first interesting strategy presented. It is one of the few strategies which does not require knowing how many output units should fire. It simply sets the threshold for each unit so that the probability of a spurious unit firing is less than some number P_{spur}, which was set at 0.001 in the tests shown. In common with most of the better strategies,knowledge of the theoretic distributions which describe the dendritic sums must be 'built in' to the thresholding mechanism. Either implicitly or explicitly, this requires that information about the parameters of the net be built into it. Most important is that this strategy sets the threshold as a function of the number of target patterns in which an output unit fires. Recall strategies that exploit unit usage achieve much better performance than the others.

The remaining strategies use the number of active input lines which impinge on an output unit in addition to other information. Marr posited that the activity information would be available to the output units in the form of inhibitory signals of magnitude $f\alpha$. Staircase and 'f only' require similar information – the dendritic sum vector, the number of units which should fire, and f (as well as being able to compute how many units will be active at particular threshold settings). As discussed above, 'f only' performs much better than staircase. Interestingly, the performance of Staircase is nearly as poor as 'T only'.

The evidence theoretic idea proposed by Gardner-Medwin makes use of the activity directly, rather than embedded in the f term. As noted above, the strategies based on this idea make use of the dendritic sums, the network parameters, and the fraction of noise bits in the cue, s. The version of this strategy which used the appropriate value of ρ_i in the calculation of the W_+ and W_- values performs much better than the one which just uses the $\hat{\rho}$ of the entire network. Again, exploiting unit usage information makes quite a difference. This strategy performs very well, however it requires the value for the fraction of spurious units in the cue in order to calculate W_+ and W_-, which is not available in practice. The two strategies based on knowing that the dendritic sum distributions of the low and high units are binomials on the activity both perform well. 'T:Pspur:a' requires knowledge of the binomial distributions for the low units only and Guess s requires knowledge of both. Again, this knowledge can be explicit (declarative) or implicitly built into the implementation (procedural). So either explicitly or implicitly, knowledge of the network parameters is required. Both strategies also require the dendritic sums, the number of desired active output units, and the ability to measure how many units will be active at given threshold settings.

It is interesting to compare the thresholds chosen by the (best) evidence theoretic strategy and Guess s. Figure 4.14 shows the thresholds that are chosen by the two strategies as a function of activity superimposed upon the clouds of simulation data for $s = .4$ and $s = .8$ ($R = 1000$, $r_i = 30$). The Guess s thresholds are computed from the expressions for the binomials and the slope of the threshold line for the evidence theoretic one is computed using the expressions in section 4.4. The intercept of this line is taken from the mean of the confidence value C selected in the simulations to get the desired number of output units firing. For $s = .4$, the thresholds chosen by these strategies are nearly identical. When $s = .8$, Guess s performs better than the evidence theoretic strategy and the Guess s thresholds are consistently higher than the evidence theoretic thresholds. The threshold line for the evidence theoretic strategy has a small positive intercept; if the Guess s 'lines' were extrapolated to low activity values, they too would have positive intercept[7]. The curving

[7]Early simulations performed without taking unit usage into consideration made it appear that lines with slope closer to 1 and negative intercept would do a good job of separating the clouds of points associated with the low and high units. Looking at the clouds for a single

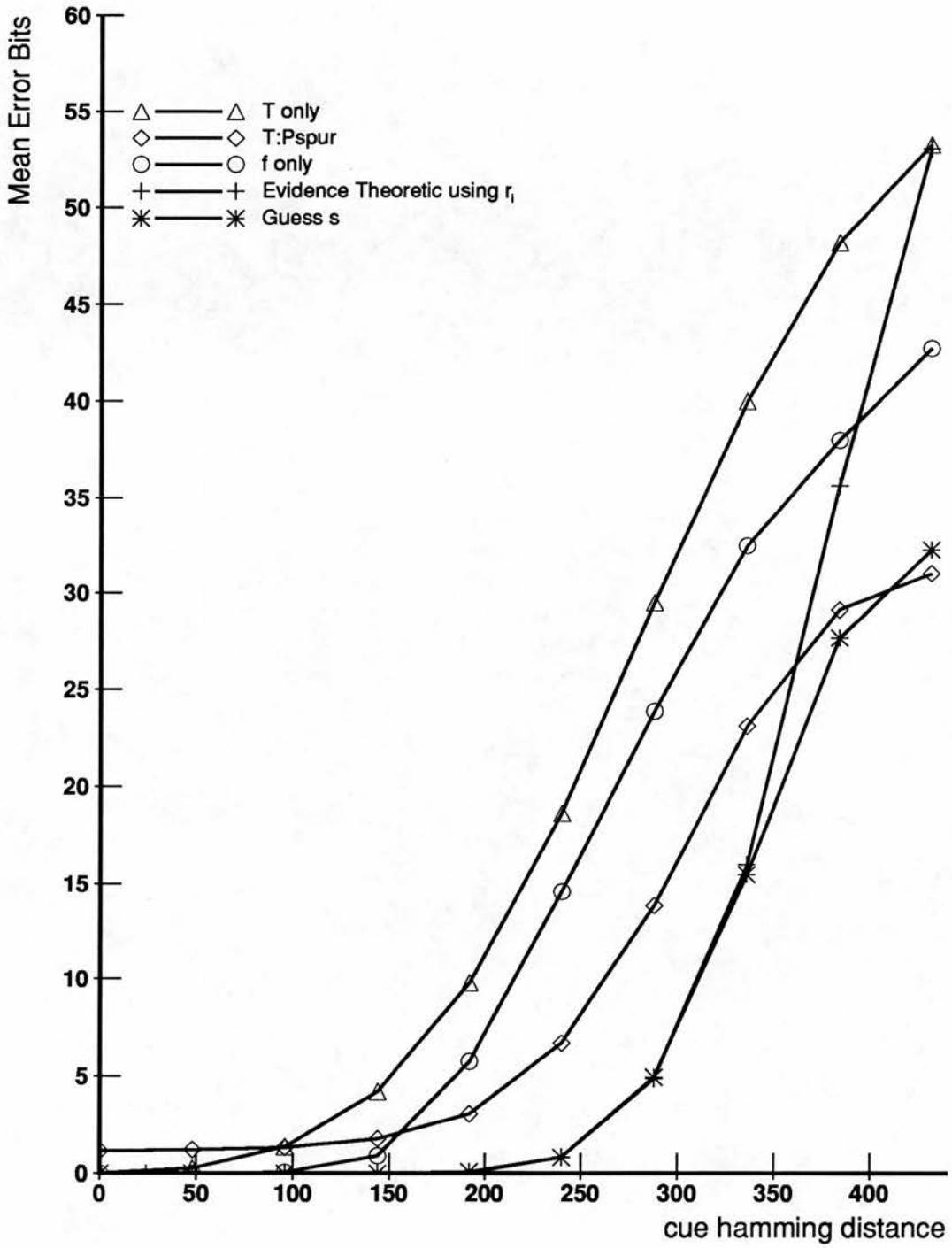

Figure 4.13: Comparison of a number of differently motivated thresholding strategies. Recall errors using noisy cues plotted as a function of the hamming distance between the cue and the input pattern. Nets defined by the canonical parameter values were used, R = 1000.

down of the Guess s thresholds as activity becomes low is really irrelevant since the dendritic sum distributions are far too coarse grained in these cases to distinguish the low from the genuine units. Since these thresholds can be described by lines with positive slope less than 1 and small positive intercepts, they are close to f threshold lines with intercept zero. This makes it clear why the omniscient 'Best f given τ_i' strategy works so well – this kind of line does a good job of separating the genuine from the low units.

In summary, we now have a thresholding strategy which we can employ in partially connected associative nets which yields good recall performance with respect to mean errors: **Guess s**. The Guess s strategy does not require knowing the answer *a priori* or knowing the noise characteristics of the recall cues. It does however exploit knowledge of the activity impinging on each unit and the number of stored patterns in which they fire. Thus thresholding based on both activity and dendritic sums have a performance advantage. The next two sections address parameter sensitivity and storage capacity of the networks using this strategy

4.6 Parameter sensitivity

We now explore how the performance of the partially connected associative net using the Guess s strategy changes as network parameters are varied. The parameters for the network employed in the extended example in this chapter were calculated using constraints originally posited by Marr (1971). This section addresses performance sensitivity to connection density, activity ratio, and to the 'exactness' of the threshold choice in the Guess s strategy. Recall performance as a function of the number of patterns stored is addressed in the next section.

value of unit usage makes the situation much clearer.

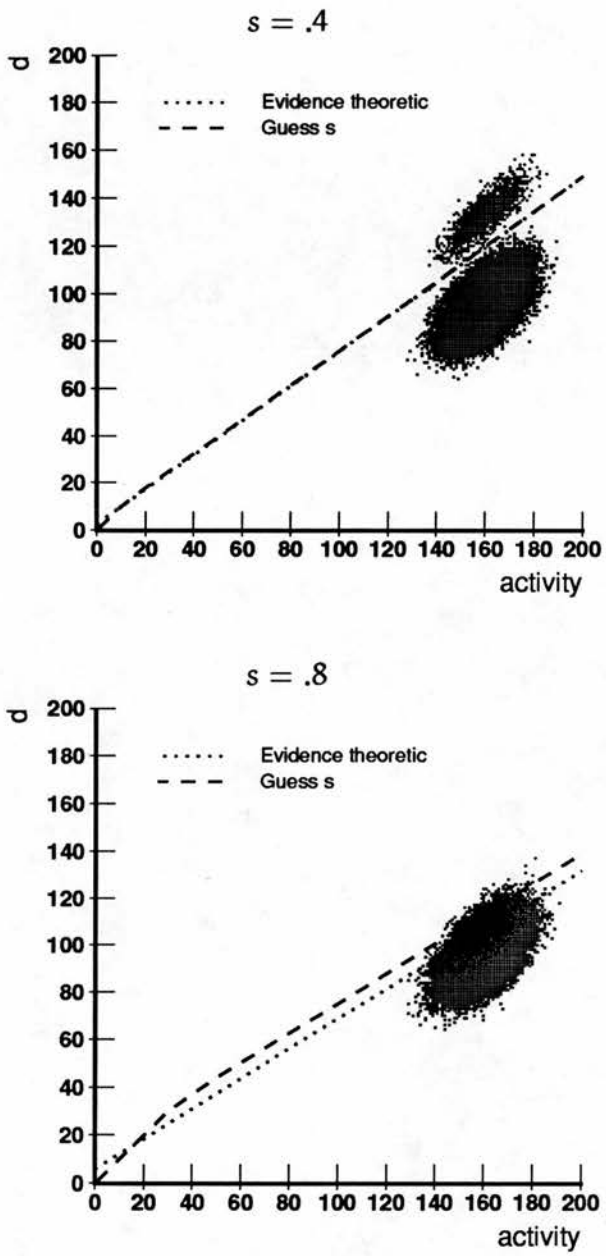

Figure 4.14: Comparing the thresholds chosen by the Guess s and 'Evidence theoretic using r_i' strategies as a function of activity. The scattergrams are of dendritic sums and activity observed in simulations. $R = 1000$, $r_i = 30$. In the top graph $s = .4$ ($m_g = 144$, $m_s = 96$). In the bottom graph $s = .8$ ($m_g = 48$, $m_s = 192$).

4.6.1 Sensitivity with respect to connection density

Connection density Z affects performance by changing the amount of activity a that reaches the output units since activity is distributed $b(m_{cue}, Z)$. At higher values of activity, the signal-to-noise ratio of the d_g and d_s distributions is greater, which reflects itself in lower probabilities of false positives and negatives.

To investigate how performance varies as a function of Z, simulations were performed using the canonical parameter set, but varying S
(S = 600, 1000, 2000, ..., 7000). The usual noisy cues were used for recall. The expected output errors for the fraction of spurious bits in a cue, $s = .5$ are calculated. Expected output error is plotted together with mean output error from simulations in Figure 4.15. The simulations conform well to the theoretical predictions. Figure 4.16 shows simulation results for various values of S against cue hamming distance; (theoretical values are not plotted because the graph would be too cluttered). As Z increases from the canonical value, performance increases, but not dramatically – the canonical parameter set was designed to yield good recall. However, as Z decreases, performance degrades; slowly at first, then it drops off sharply as Z becomes less than 0.5. Still, even at S = 600, performance using full cues is good. At this connection density, the expected activity when using a full cue is 18, which is about as low as it can go if reasonable recall performance is to be expected.

4.6.2 Sensitivity with respect to pattern activity ratio

The coding of the input and output patterns has a critical effect on the performance of associative nets as we saw in Chapter 3. Sparse patterns, that is, patterns with $M \simeq \log_2(N)$, yield optimal information efficiency in the fully connected associative net (Willshaw, 1971). However, sparsely coded patterns do not lend themselves to recall using partial patterns. If content-addressability is required, the number of active bits must be greater than $\log_2(N)$. In the partially connected associative net, the number of bits active in the input patterns must also be greater than required in a fully connected net in order that an acceptably

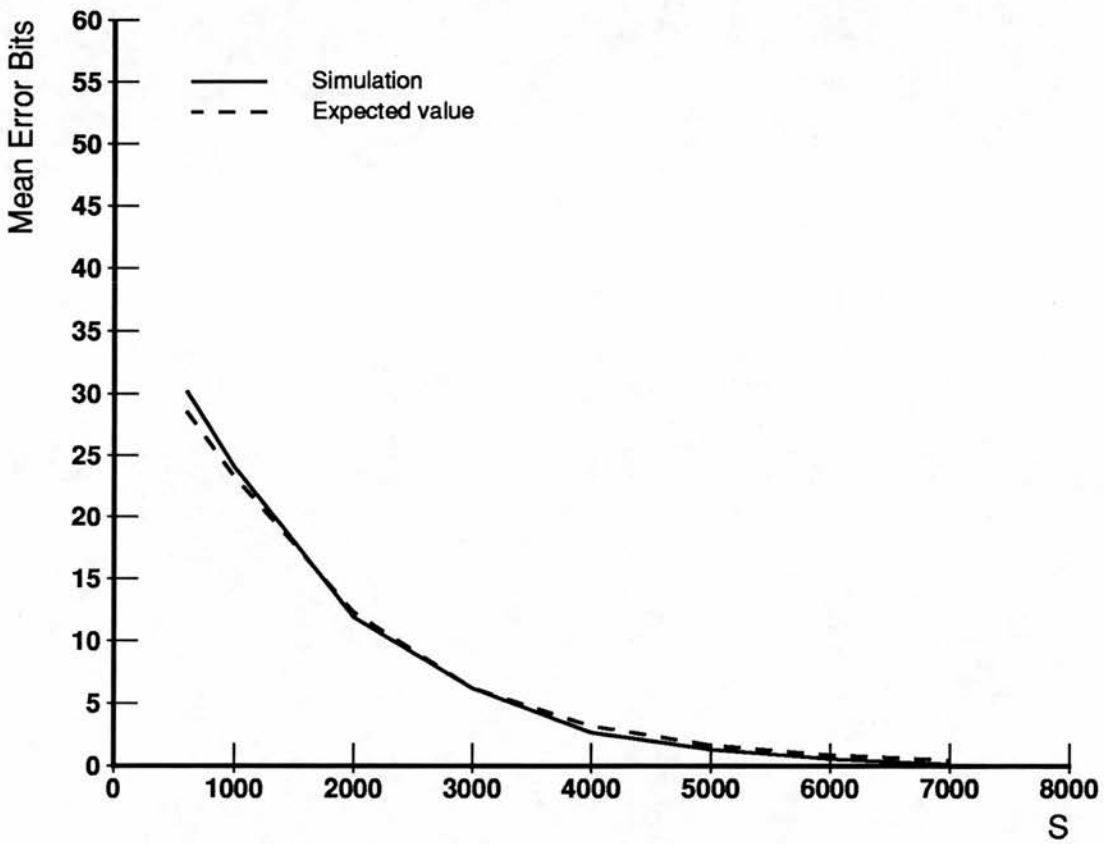

Figure 4.15: Parameter sensitivity: the number of synapses S on an output unit. Comparing theoretical expected error with simulation recall errors as a function of the number of synapses on an output unit, S = 600, 1000, 2000, ..., 7000, for cues with $s = .5$ ($m_g = m_s = 120$).

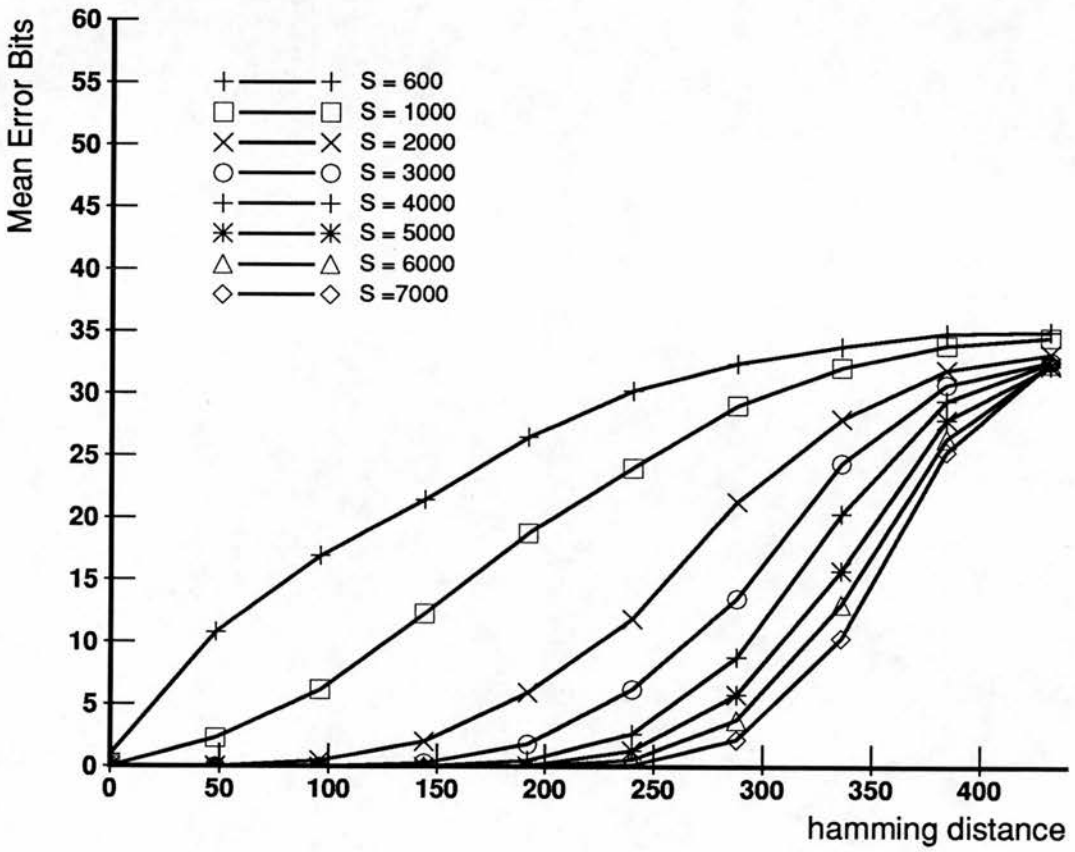

Figure 4.16: Parameter sensitivity: the number of synapses S on an output unit. Comparing recall errors for S = 600, 1000, 2000, ..., 7000. Recall errors using noisy cues plotted as a function of the hamming distance between the cue and the input pattern.

high number of active input lines actually contact synapses on the output units. In the canonical example net, the activity ratio of the input patterns is set to a value appropriate for recall using partial cues that have approximately 10% of the genuine bits active. If the coding is more sparse, recall performance using partial cues will degrade, but more pattern pairs can be stored and still achieve good recall using full cues.

The values of the activity ratios α_A and α_B directly affect the proportion of modified synapses ρ_i, which has proved to be one of most critical network attributes. α_A and α_B can be varied independently or it can be assumed that they are equal. This assumption has been made up till now in this thesis. In Figure 4.17, $\hat{\rho}$ is plotted as a function of α for R = 1000. Two curves are plotted; in the first, one of the α is held fixed at .03 and the other α is varied and in the second curve $\alpha_A = \alpha_B$. In both, we see that ρ increases sharply with α. Thus network performance is very sensitive to changes in pattern coding. If one has control of this coding, patterns should be coded as sparsely as possible with respect to the degree of content addressability required. For good recall to occur, the condition $m_g Z \geq 20$ needs to be satisfied for the genuine units which constrains the sparsity of connections and/or the number of genuine bits in the cue.

Simulations were performed to test how α affects recall performance. Simulations were run with input and target pattern sets with

$$\alpha_A = \alpha_B = \ln(N)/N, .01, .02, .03, .04, 05.$$

Results are shown in Figure 4.18

4.6.3 Sensitivity with respect to threshold choice

The Guess s strategy is designed to find the thresholds that will minimize the expected number of false positives and negatives in the output pattern. How sensitive is recall performance to perturbations in its thresholding? If the threshold is higher than the one that minimizes the expected number of false positives and negatives, fewer false positives and more false negatives would be expected. Similarly, if the threshold used is too low, more false positives

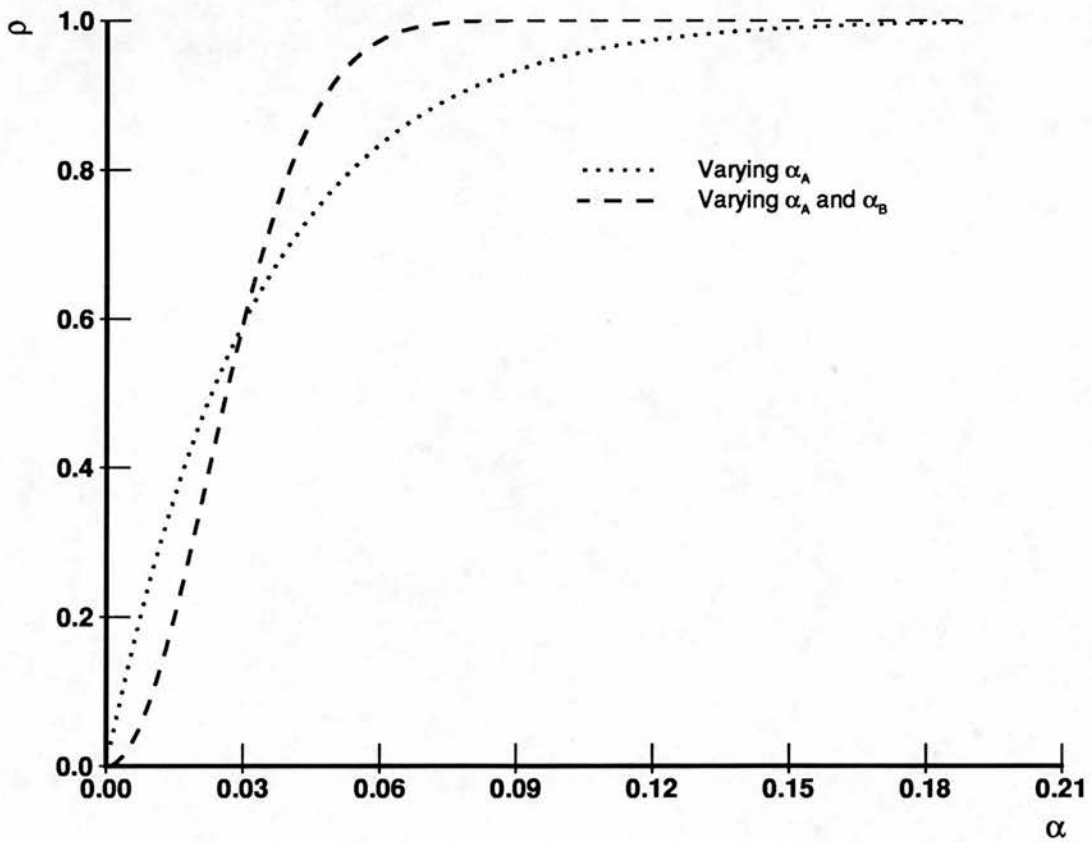

Figure 4.17: The expected proportion of modified synapses, $\hat{\rho}$, as a function of the input and output activity ratios, α_A and α_B. In the first curve, one of the α is held fixed at 0.03 and the other varies; in the second curve, $\alpha_A = \alpha_B$. Canonical parameter values were used in the calculations, and R = 1000.

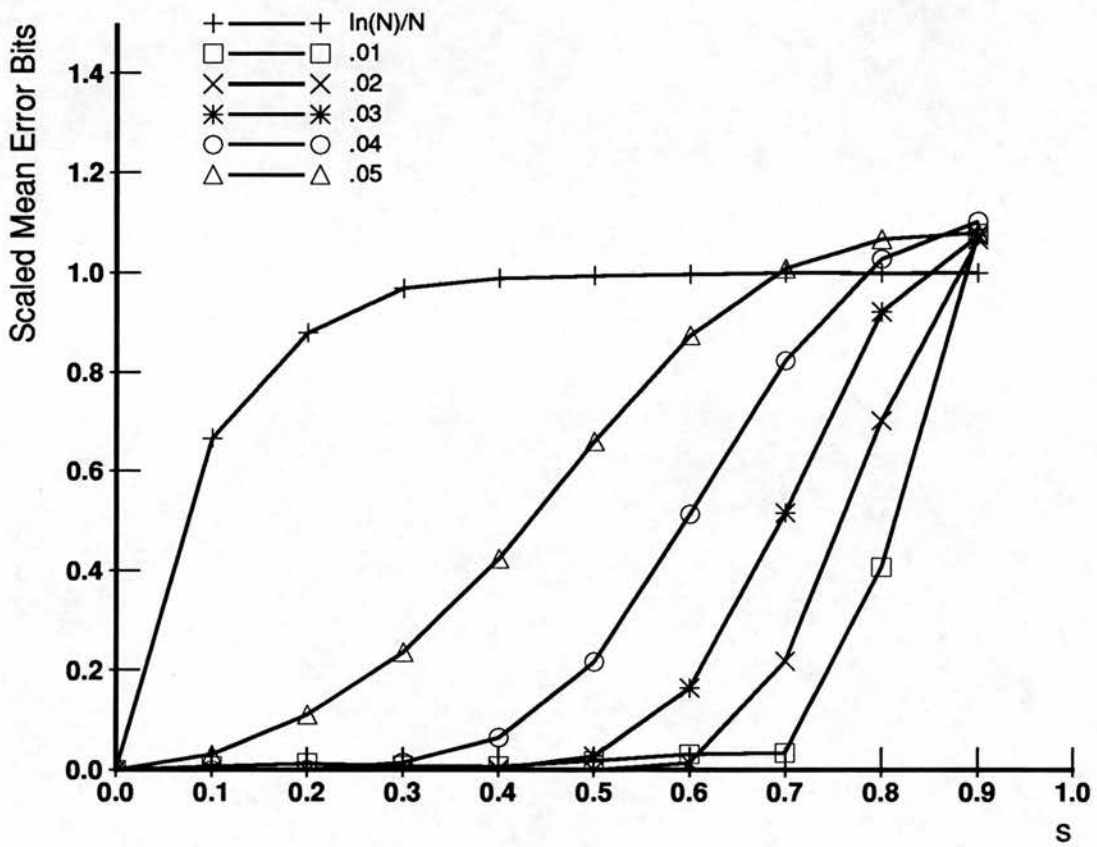

Figure 4.18: Parameter sensitivity: α. Comparing recall errors for pattern sets with $\alpha_A = \alpha_B = \ln(N)/N$, .01, .02, .03, .04, 05. Since the output patterns have varying numbers of desired active units, the mean output error is scaled – the vertical axis is $\langle\Delta\rangle/M_B$. Scaled mean errors is plotted as a function of the fraction of spurious bits in the cue, s. Nets defined by the canonical parameter values were used, $R = 1000$.

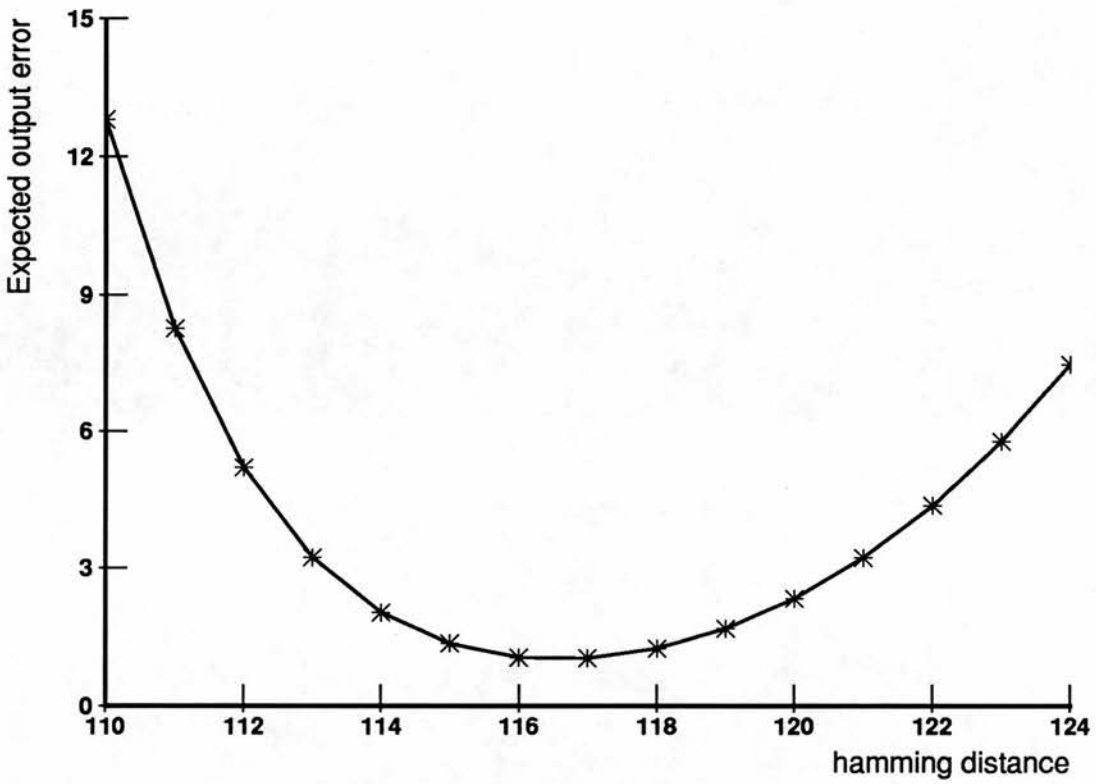

Figure 4.19: The expected output error as a function of the threshold T in the Guess s strategy. Values are calculated for $a = 160$, $r_i = 30$, $s = .5$. In this case, Guess s selected a threshold T = 117.

and fewer false negatives are likely. If the measure of the activity of a unit that reaches the thresholding machinery is inaccurate, either case could occur. In this section, the performance is investigated when noise is added to the 'correct' threshold and when noise is added to the activity value used in calculating the appropriate threshold.

To gain an initial view of how performance is affected by threshold choice, Figure 4.19 shows the expected output error as a function of the threshold t for $a = 160$, $r_i = 30$, $s = .5$. In this case, the threshold that minimizes the expected output error is 117. It can be seen from this figure that small deviations in the threshold choice should not affect recall errors too adversely.

To test the sensitivity of the performance of Guess s to the threshold chosen, sim-

ulations were performed in which the result of the threshold selection function is systematically perturbed. Experiments were performed in which random noise from gaussian or Poission distributions was added to t. With the gaussian noise, the threshold returned can be higher or lower than that which minimizes expected output error. With Poisson noise it can be higher, but not lower. In the experiments, relatively small amounts of noise were added to the Guess s threshold value since large amounts of noise would clearly result in poor recall performance.

To test how sensitive the algorithm is to the accuracy of the activity measure, experiments were performed in which gaussian noise was added to the activity value sent to the threshold selection function; the canonical parameter set was used and $R = 1000$. Results are shown in Figure 4.20. They show that as long as the amount of random noise added to the threshold or activity measure is small, recall performance is not much different from the exact algorithm.

4.7 Storage capacity and information efficiency

One important question is "How many pattern pairs can these networks store." The number of pattern pairs that can be stored depends on the recall task and the performance criteria. However, the number of pattern pairs that can be stored with good recall does does not tell the whole story. Output vectors with different N and M require different amounts of information to describe them and nets can have different numbers of synapses. Information efficiency is also of interest.

This section explores the storage capacity and information efficiency of partially connected nets using the Guess s thresholding strategy. First, the recall performance of the canonical net is shown as a function of the number of pattern pairs stored. Next, the criteria for good recall is stated. The storage capacity and information efficiency of the canonical net are then presented. Then, the storage capacity and information efficiency of this class of net are investigated as functions of the number of units in the net, of connection density, of activity level, and of the number of synapses in the net.

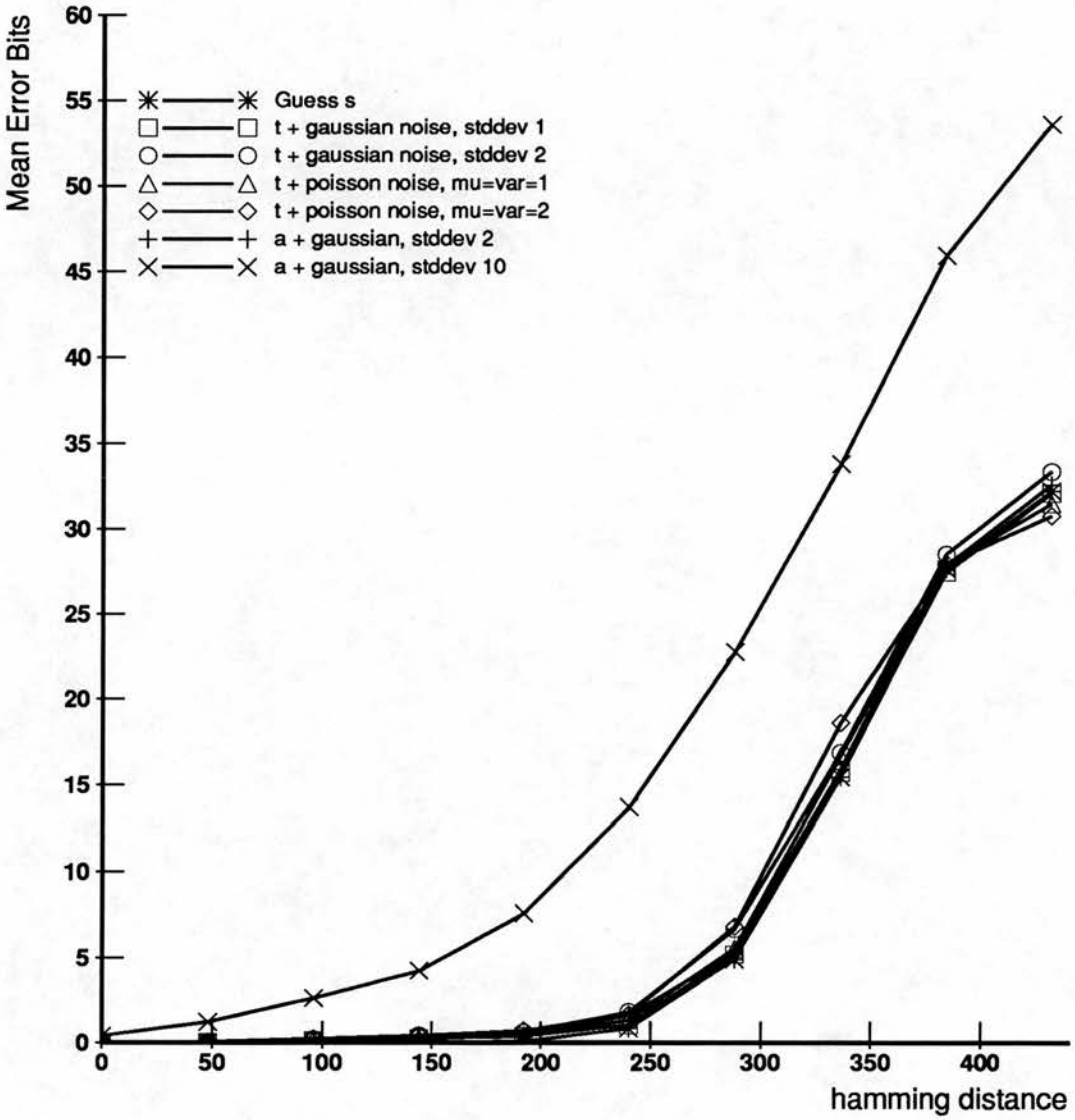

Figure 4.20: Algorithm sensitivity: Comparing recall errors for versions of the Guess s algorithm in which random noise was added to the threshold choice or the activity measure. Recall errors using noisy cues plotted as a function of the hamming distance between the cue and the input pattern. In the gaussian noise cases, noise values were taken from a gaussian distribution with mean 0 and standard deviation either 1 or 2. For Poisson noise, values were taken from Poisson distributions with means of 1 or 2.

4.7.1 Performance as a function of R

To gain an overview of how the performance of the partially connected net using Guess s varies with the number of pattern pairs stored, Figure 4.21 plots mean output errors from simulations as a function of R for noisy cues. Mean output error increases as the number of pattern pairs for all cues, but it increases much faster for the noisier cues.

4.7.2 Storage capacity of the canonical net

The number of pattern pairs that can be stored depends on the recall task and the performance criterion. Let the criterion for 'good recall' be that the mean or expected number of bits in error in the output pattern is less than or equal to 1.

The information efficiency of this net can be calculated as in Chapter 2. Information efficiency is defined to be the total amount of information (in bits) which can be recalled from the net divided by the number of bits of storage required by the net. In addition to the usual synapses on an output unit, the synapses required by the activity measuring machinery must be considered when totalling up the amount of storage required. A worst case situation is that in which each output unit has an inhibitory unit associated with it which measures the activity impinging on the output unit by having contacts (with weight 1) from each input unit that contacts the output unit in question. That is, twice as many synapses are required. For the moment, the small amount of storage that each unit requires to hold unit usage information is neglected.

Call the amount of information required to describe one of the output patterns I_0. For the canonical net,

$$I_0 = \log_2 \binom{N_B}{M_B} = \log_2 \binom{1024}{30} = 191.67$$

Let information efficiency be given by

$$\eta = \frac{R_{good} I_0}{2 N_B S}$$

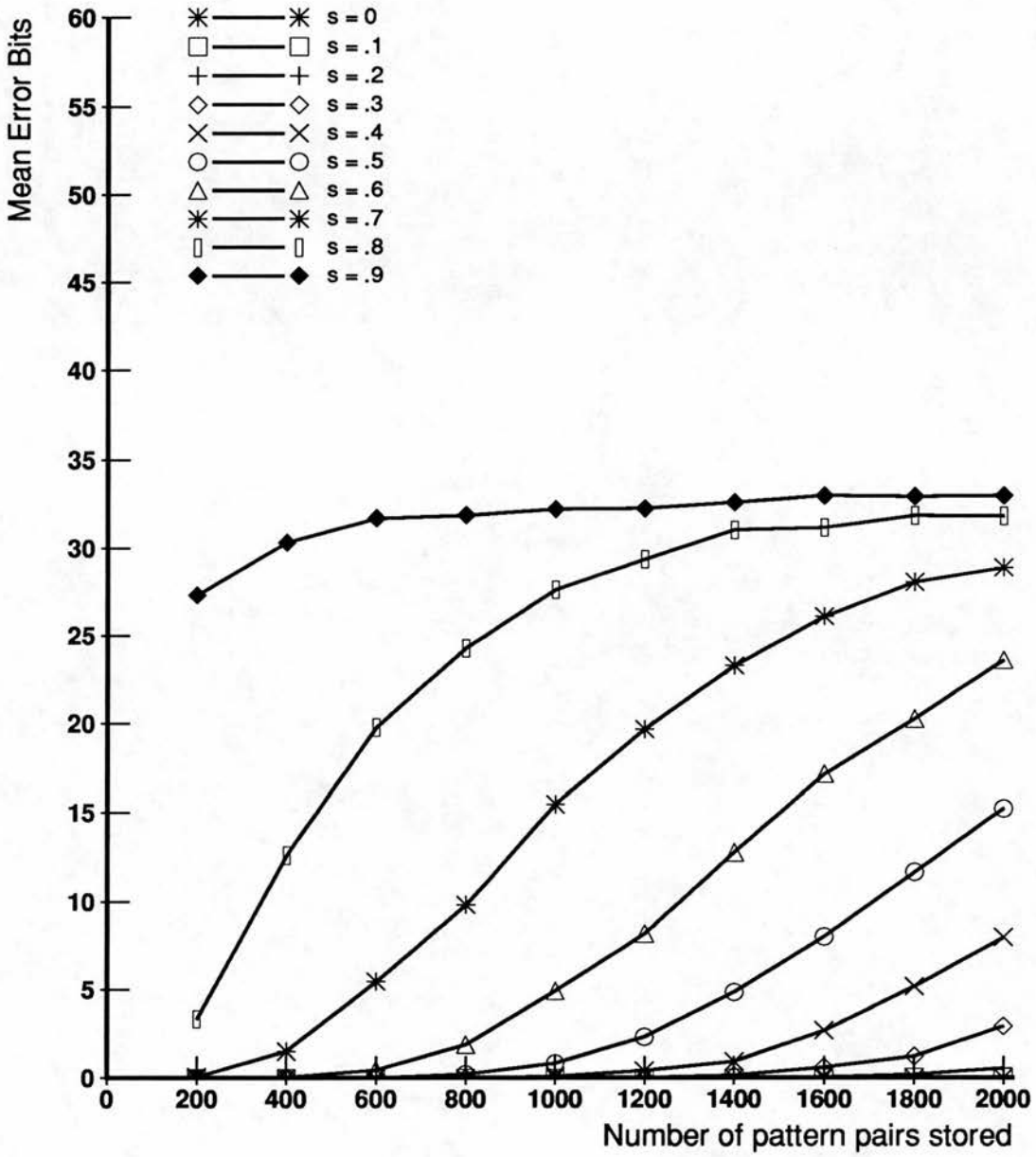

Figure 4.21: Performance as a function of R. Mean output error from simulations is plotted as a function R for the usual noisy cues. The graph shows one curve for each value of s, the fraction of noise bits in the cue. (The number of noise bits equals the number of missing bits.) Canonical network parameter values are used in the simulations.

where R_{good} is the number of pattern pairs that can be stored such that recall performance is good as defined above.

This section discusses the storage capacity and information efficiency of this net for three cases:

1. Using full cues.
2. Using partial cues with ten percent of the genuine bits active.
3. Using noisy cues with half genuine, half noise bits

Using full cues, the expected number of recall errors was calculated for increasing values of R using the canonical parameter values. At $R = 3200$, $E[\Delta] = 1.0396$ – it just edges above 1. Thus 3200 pattern pairs can be stored in this net such that recall performance is still good. This gives $\eta = .056$.

The expected number of recall errors using cues with 24 bits active out of the 240 in the stored input pattern was calculated for increasing values of R using the canonical parameter values. At $R = 1000$, $E[\Delta] = 1.17$, and from simulations $\langle\Delta\rangle = .949$. Consequently $R_{good} = 1000$ and $\eta = .0175$ for partial cues with ten percent of the genuine bits active.

For noisy cues with 120 genuine bits and 120 spurious bits active expected recall errors were calculated. At $R = 1000$, $E[\Delta] = 1.34$ and from simulations $\langle\Delta\rangle = .82$. Again, $R_{good} = 1000$ and $\eta = .0175$.

These information efficiency figures are not terribly impressive. However, the network parameters were constructed in order to provide a useful illustrative example for this chapter. Comparing information efficiency of this net with that of the corresponding fully connected associative net with the same parameter values for N and M may be useful. Consider a fully connected associative net defined by $N_A = 8000$, $N_B = 1024$, $M_A = 240$, and $M_B = 30$. Using the same performance criteria, ($E[\Delta] \leq 1$), it can store 3600 pattern pairs using full recall

cues. From Chapter 2, its information efficiency is

$$\eta \simeq \frac{R_{good} I_0}{N_A N_B} = 0.084$$

Thus the information efficiency of the partially connected net is not much worse than that of the fully connected associative net in this case. Note that if the activity measuring machinery of the net requires less than one synapse for each synapse on the output units, the information efficiency of the partially connected net will be correspondingly better.

Thus the number of pattern pairs that can be stored and achieve some performance criteria in terms of recall errors can be calculated. How many patterns for a particular net depends on the recall task and the definition of 'good recall'. For a given definition of good recall, if content-addressable recall is required using partial or noisy cues, then fewer pattern pairs can be stored than if simply associative recall with full cues is required. Given a memory task, the parameters of the net can be constructed so that the net will achieve the performance required.

We now know the storage capacity and information efficiency of the canonical net. The question that now arises is how these change when networks parameters are varied. In the next few sections, the R_{good} and η are calculated for recall from full cues for nets with different numbers of units, different connection densities, and different activity levels.

4.7.3 Storage capacity and information efficiency as a function of N

We first consider how the storage capacity and information efficiency change as the number of units increases. Let us consider 'square' nets ($N_A = N_B$ and $M_A = M_B$). If α does not vary, then for a particular value of R, $\hat{\rho}$ is the same no matter how many units are in the net. If Z is kept constant as well, MZ increases with N so performance will improve, but not dramatically since $\hat{\rho}$ does not change. The amount of information required to describe an output pattern also increases nearly linearly with N (if α is not changed), but the amount of

storage increases as N^2.

R_{good} and η are calculated for nets with increasing numbers of units. Results presented earlier in this chapter show that these nets perform best when α is low and Z is high. However, high connection density implies that more storage is required for the net. For these calculations, $\alpha = .01$ and $Z = .1$. Figure 4.22 shows R_{good} and η as functions of N. We see that as N increases, R_{good} increases steeply at first, then much more slowly. Information efficiency is maximal at $N = 40000$ for these values of the parameters α and Z.

4.7.4 Storage capacity and information efficiency as functions of Z and α

This section considers how the storage capacity and information efficiency of a net with a fixed number of units change with the connection density and activity level. Results presented earlier in this thesis show that recall performance improves as connection density increases. The amount of storage required for the net increases linearly with Z, so the question for information efficiency is "Which increases faster, R_{good} or NS?" As the activity level is increased, $\hat{\rho}$ increases so R_{good} will decrease rapidly. I_0 increases with α, but not as fast as R_{good} decreases, so information efficiency will decrease.

R_{good} and η are calculated for nets with different values of Z and α. Since recall in these networks becomes poor when MZ is low, a large net is used for the calculations: $N = 320000$.

In section 4.6.1 we saw that performance is good across a wide range of connection densities so we would expect the information efficiency to be a relatively flat function of Z. Figure 4.23 shows R_{good} and η as functions of Z; in this figure, $M = 1000$. R_{good} increases steeply at first, then much more slowly as Z increases. η is maximal at $Z \simeq .031$ where $\eta = .14$. Indeed, it is relatively flat as Z decreases until Z goes below .02 when is drops sharply.

Figure 4.24 shows R_{good} and η as functions of α. R_{good} was calculated for a net

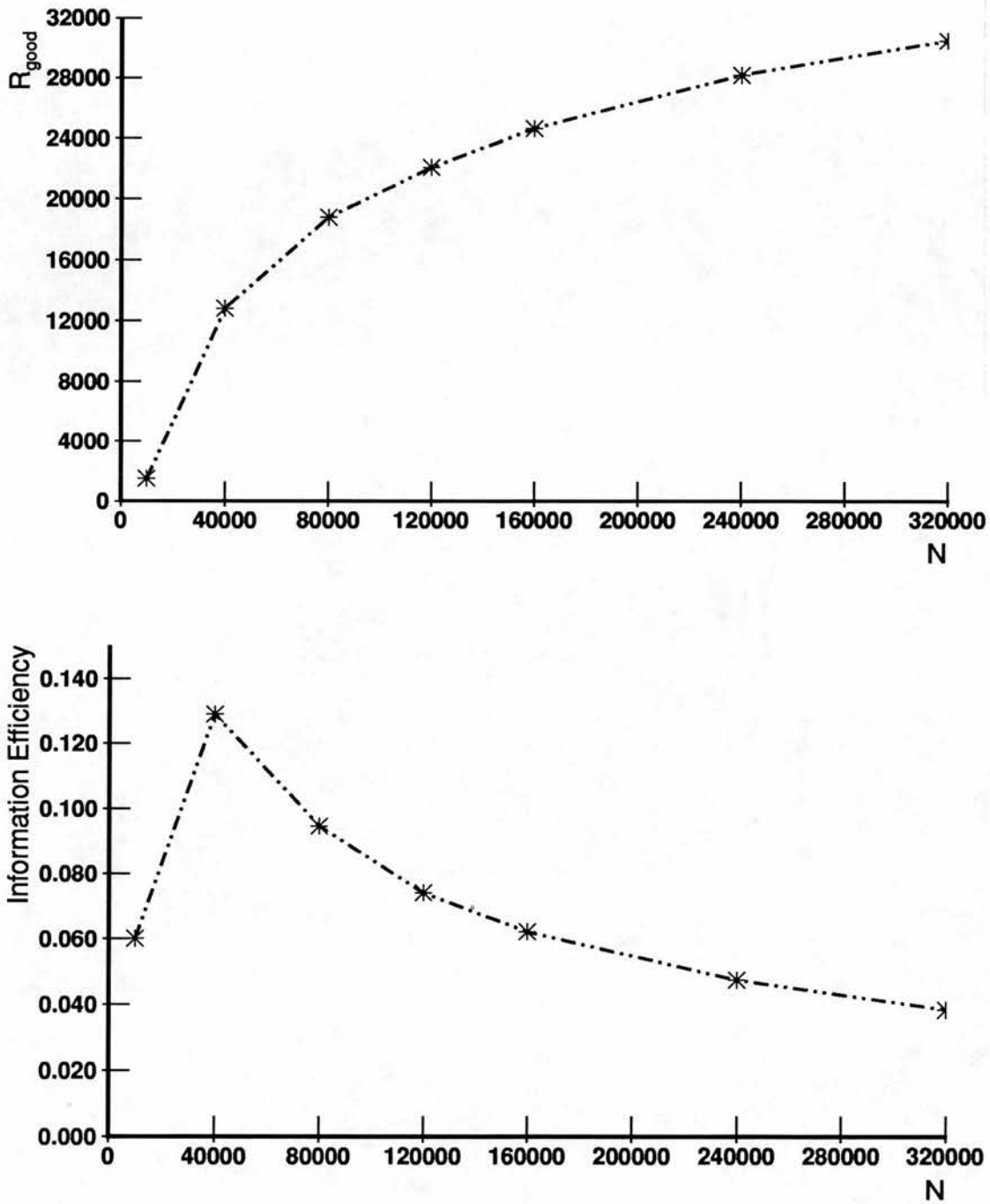

Figure 4.22: R_{good} and η as functions of N. The top graph shows R_{good} as a function of N and the bottom one η as function of N. R_{good} is obtained by calculating the number of pattern pairs that can be stored in the net and obtain good recall ($E[\Delta] \leq 1$). For these calculations, $\alpha = .01$ and $Z = .1$.

defined by $N_A = N_B = 320000$ and $S = 16000$. $M_A = M_B$ were varied to vary α_A and α_B. As α increases, R_{good} decreases. It drops sharply at first, then more slowly. Information efficiency is maximal at $\alpha = .002$ where $\eta = .1366$ for this net.

Note that in both of these cases, information efficiency is maximal when $MZ \simeq 32$.

4.7.5 Information efficiency for a fixed number of synapses

Given a fixed number of synapses, from an information efficiency point of view, what is the best way of utilizing them? If the number of synapses is held constant, it must be decided what other parameters will be varied or held constant. We can vary N, M, and S subject to the constraint that NS is a constant. In this case, to maximize information efficiency, we want to maximize $R_{good}NI_0$. Considering each of the terms of this expression in turn, R_{good} tends to increase as Z increases and as α decreases. If NS is constant, as N increases Z decreases. I_0 is maximum at $\alpha = .5$, and in the range of α in which these nets function best (low α), it is nearly linear with α.

To address the question of the best way to utilize a fixed number of synapses let us consider an example. Again, let us consider a 'square' net such that $N_A = N_B$ and $M_A = M_B$. Let $NS = (320000)(16000) = 5120000000$. The connection density, Z, will be varied, and at each value of Z the information efficiency will be found for several values of M. In Figure 4.25, the value of η at $\alpha = .002$ and the maximum of the η calculated are plotted as a function of Z. The maximal η increases with Z. If α is fixed as it has been throughout most of this thesis, then η is best for more sparsely connected nets. For these calculations, the values of M that are used are such that MZ varies from 20 upwards. Though the search was not exhaustive, these data suggest that if α can be varied to improve information efficiency, η, is maximal for this example when MZ is near 32 for low Z and near 20 for higher Z. This is in agreement with the results of Marr (1971) and Gardner-Medwin (1976).

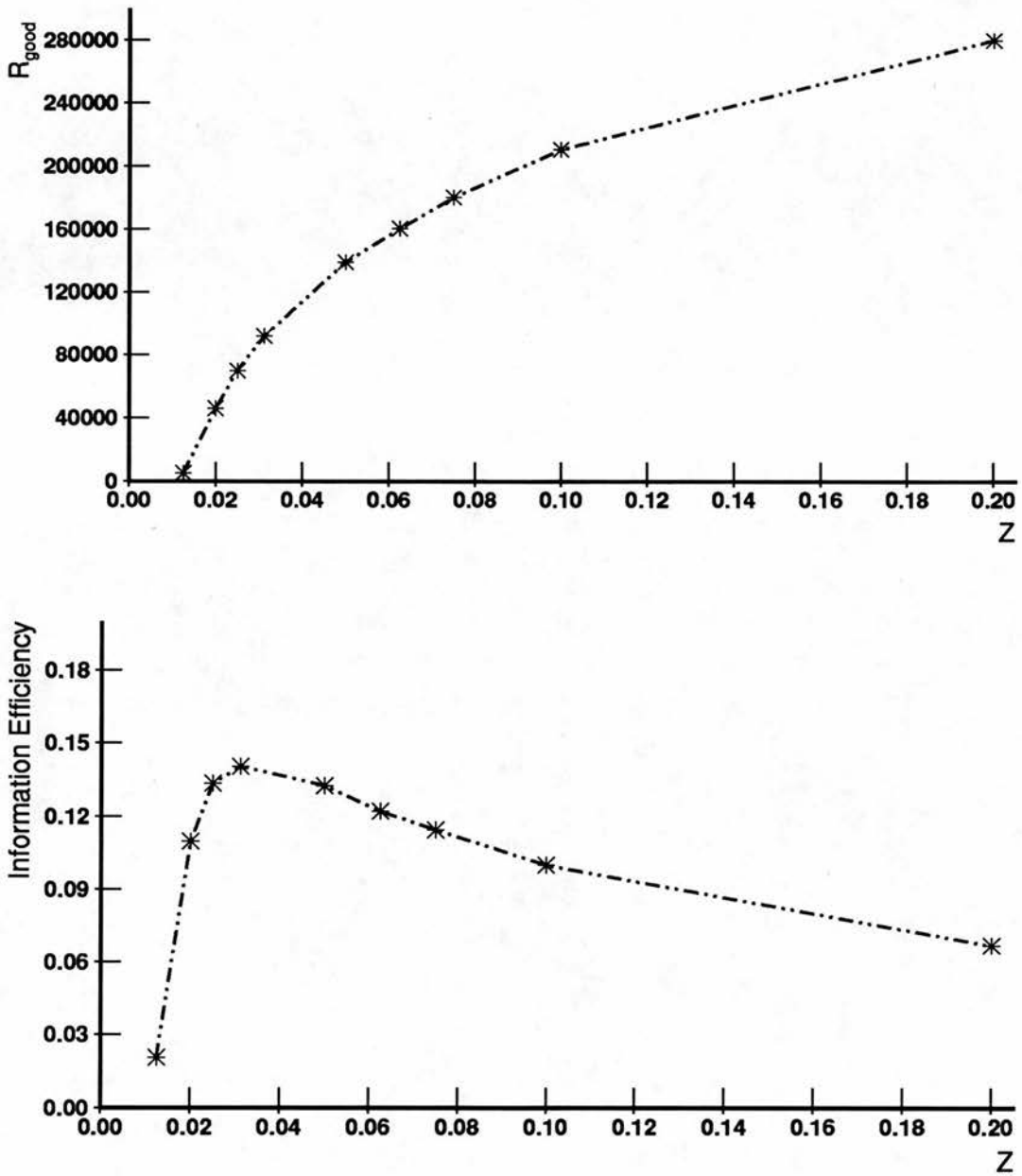

Figure 4.23: R_{good} and η as functions of Z. R_{good} is obtained by calculating the number of pattern pairs that can be stored in the net and obtain good recall ($E[\Delta] \leq 1$). Parameter values used in the calculations: $N_A = N_B = 320000$, $M_A = M_B = 1000$. S is varied to vary Z.

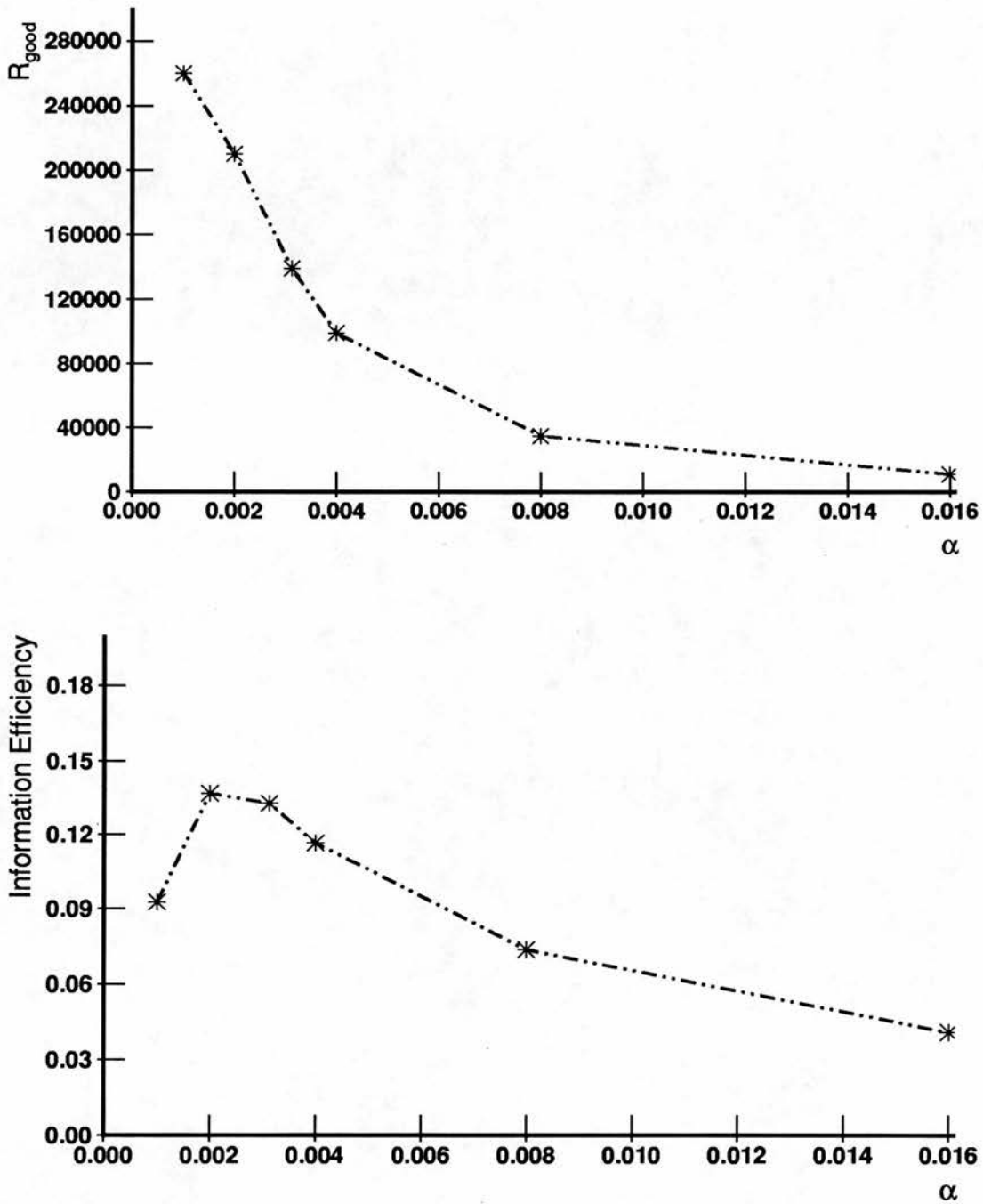

Figure 4.24: R_{good} and η as functions of α. R_{good} is obtained by calculating the number of pattern pairs that can be stored in the net and obtain good recall ($E[\Delta] \leq 1$). Parameter values used in the calculations: $N_A = N_B = 320000$, $S = 16000$. $M_A = M_B$ is varied to vary α.

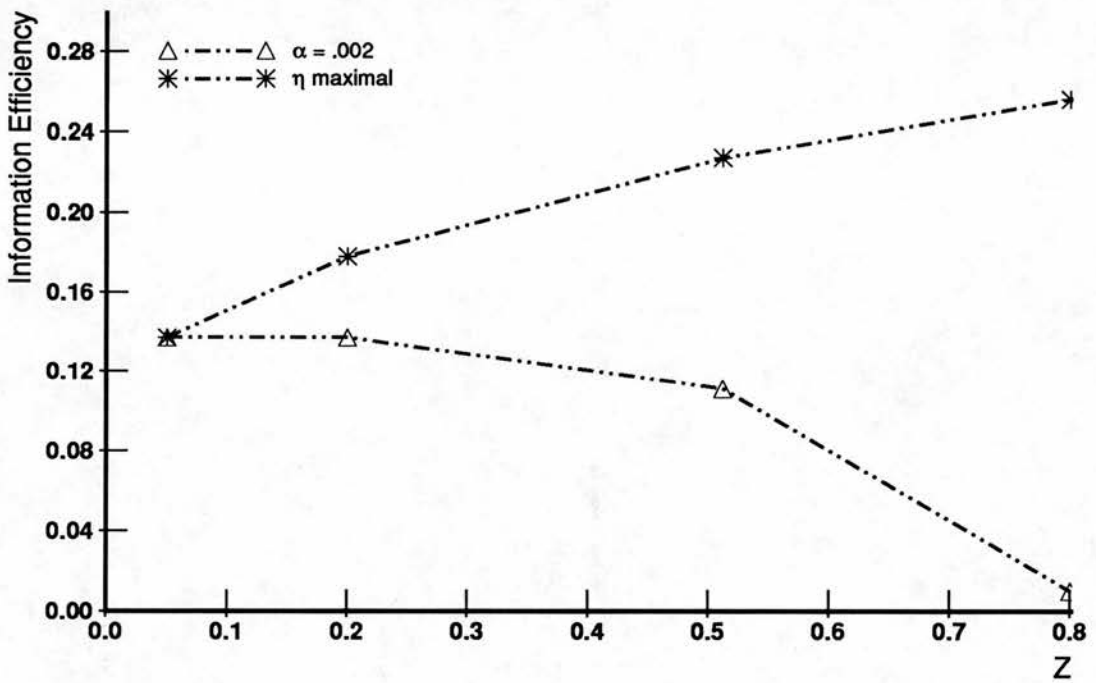

Figure 4.25: Information efficiency as a function of Z for a fixed number of synapses. It is obtained by calculating the number of pattern pairs that can be stored in the net and obtain good recall ($E[\Delta] \leq 1$). Parameter values used in the calculations: $N_A = N_B = 320000$, $S = 16000$. $M_A = M_B$ is varied to vary α.

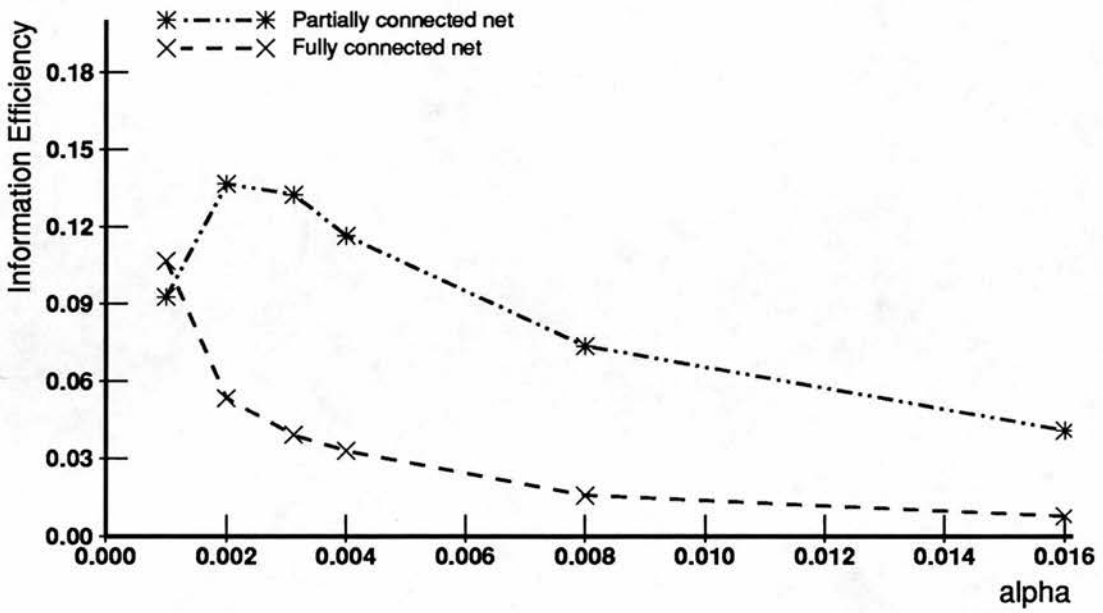

Figure 4.26: Comparison of information efficiency of the partially and fully connected associative nets. R_{good} is obtained by calculating the number of pattern pairs that can be stored in the net and obtain good recall ($E[\Delta] \leq 1$). Parameter values used in the calculations: $N_A = N_B = 320000$, $M_A = M_B$ is varied to vary α. For the partially connected net $S = 16000$.

4.7.6 Comparing information efficiency of the partially and fully connected associative nets

Now that we have some impression of how information efficiency changes with changes in network parameters, the question arises "How do the partially and fully connected associative nets compare with respect to information efficiency?" In the case of the canonical example net used in this thesis, information efficiency of the fully connected net is somewhat higher than that of the partially connected one. Let us consider information efficiency in the larger net used in the last section. Figure 4.26 shows η as a function of α for the fully and partially connected nets. As above, $N_A = N_B = 320000$; for the partially connected net $S = 16000$. We see that information efficiency of the partially connected net is better than that of the fully connected net except for the case of $\alpha < .002$. As expected, the information efficiency of the fully connected net improves steadily as α decreases.

4.8 Discussion

The question addressed in the beginning of Chapter 3 was how well can the partially connected associative net perform in the associative and content-addressable memory tasks. The work presented in this chapter has shown that they can perform well when certain thresholding strategies are used.

Given a network in which a number of pattern pairs have been stored, how, during recall, can the units which should fire be distinguished from those which should remain quiet? An obvious starting point is to perform a thresholding operation on the dendritic sums of the output units, so in Chapter 3 the distributions of these sums for the low and genuine units were studied. In that chapter, thresholding strategies based on these distributions were developed. Recall performance of the net using these strategies was not impressive. Marr suggested that primary cells in a memory structure may have access to some measure of the number of active input lines impinging on them and that this activity measure could be exploited in threshold setting. This has been the subject of this chapter. The dendritic sum distributions of the low and high units as a function of this input activity were studied, and it was found that they are simple binomials on activity. There are many ways that this activity could be used in threshold setting. Marr suggested several, Gardner-Medwin suggested the application of evidence theory, and the finding that the dendritic sum distributions are binomial suggested others. These suggestions were turned into working thresholding strategies so that their actual behaviour could be studied.

Several of the strategies enable the partially connected associative net to function well as an associative and content-addressable memory. The best strategies exploit information about the activity impinging on an output unit and number of patterns in which each unit should fire. It was found that the threshold setting strategy developed in this thesis, **Guess s**, does a good job of distinguishing the output units which should and should not fire. The evidence theoretic strategy does not perform as well when very noisy patterns are presented, but given further work it could probably be improved.

With a good thresholding strategy in hand, the parameter sensitivity and storage

capacity of the partially connected associative net were studied. It is found that performance is not affected too much by small changes in connection density or the accuracy of the thresholding mechanism, but changes to α make a large difference.

Since Guess s is derived directly from the binomial distributions which describe the dendritic sums, analytic statements about the expected recall errors which arise when using this strategy can be made. This makes it possible to describe the recall performance of networks much larger than those which can be simulated using currently available computing resources. Though large by current simulation standards, the example net which has been used in this thesis is small with respect to mammalian nervous system structures.

Still, the example net has been very useful. Early experiments with it motivated the investigation of how unit usage affects the dendritic sum distributions. In pattern sets in which the genuine units are selected at random (such that a fixed number are active in each pattern), the number of times each unit fires across all patterns comes from a binomial distribution. Most capacity analyses of associative nets are made for this kind of random pattern, but the fact that the units take part in different numbers of patterns is not considered, leading to overestimates of storage capacity. This example net has also helped to motivate and verify analytic statements about the dendritic sum distributions, which inspired some of the better thresholding strategies. Study of simulation results have illuminated why the strategies perform as they do.

canonical example

Though the information efficiency of the partially connected net is low, it is not much different from that of a fully connected associative net of that size (in terms of the N and M) and the storage capacity for a net of this size is good – it is just a bit less than that of the fully connected net. Information efficiency is better in larger nets in which the connection density and pattern coding can be made much more sparse and still satisfy the constraints which govern their operation. Indeed, in some cases, like the larger example shown above, information efficiency of the partially connected net can actually be better than that of the fully connected associative net of the same size.

So what can be concluded about the partially connected associative net? When

should one be used in an application in place of a fully connected one? A few points about the fully connected associative net should be considered. It works most efficiently with very sparsely coded patterns. The threshold rule is simple and clean. It only requires the dendritic sums and a measure of the number of active bits in the cue. There is no need to worry about the tradeoff between false positives and negatives in the output. The way the associative net is used all the genuine units in the target pattern *do* fire. This means that false positives are the only source of error. It really does require fully connectedness because the genuine units will fail to fire if its dendritic sum is less than the number of active input lines. Thus the dendritic sum measurements and measure of the number of active bits in the cue must be quite accurate. It can perform recall from partial cues, but not from noisy ones. Willshaw (1971) suggested that recall from noisy cues could be done if the net could threshold on the number of genuine bits in the cue. This would require a much more complex threshold setting mechanism than the elegant and simple one it has. Actually, it would be just Guess s for one value of input activity.

The partially connected associative net generally requires patterns with more active bits than the fully connected one. If the patterns which must be stored have many more than $\log_2(N)$ active bits, the partially connected net can be more informationally efficient than the fully connected one. The threshold setting strategy requires more machinery than that employed in the fully connected associative net, but this enables it to deal with missing synapses, noisy input, and perturbations to the thresholding mechanism.

The work presented in this chapter supplies the tools needed to support the design decisions in implementing a device for a given memory task. Given the characteristics of the input and target patterns (N and M), the criteria for good recall, the memory task in terms of what kind of cues will be used for recall, and the number of patterns to be stored, the connection density can be calculated. Of course, it may be that the desired number of patterns cannot be stored in a single-layer net, in which case more complex architectures are required for the task, such as multi-layer nets.

With reference to the original motivation for this thesis, the pyramidal cells in most brain regions, including the hippocampus, are not fully connected, and

so understanding the properties of partially connected networks may provide part of the foundation for understanding how these brain regions work.

Chapter 5

Self-Organizing Nets

Chapters 3 and 4 present a characterization of the partially connected net in the associative and content-addressable memory tasks. As Marr noted (1971), these nets can also be used in a self-organizing mode. In this case, target vectors are not specified *a priori*, but are developed during and by the training process. In essence, these self-organized target vectors are new representations of the input vectors. After training, the goal is the same as that addressed in Chapters 2 and 3: good recall in the associative and content-addressable memory tasks.

Given an input pattern to be stored, Marr argues that the output units which should be active in the representation of that vector are those which have the greatest number of active afferents, and he suggests that this may be biologically plausible. This chapter is about one mechanism for constructing such representations. This mechanism is defined and then, using the representations it constructs, the performance of the net in the associative and content-addressable memory tasks is evaluated.

5.1 Construction of the output layer representation

Marr (1971) suggests that the output units which should be active in the representation of that vector are those which have the greatest number of active

afferents. He suggests that the synapses from the input units onto the output units are Brindley synapses, which have a small unmodifiable excitatory component w_0 (Brindley, 1967). Given a network in which no patterns have yet been stored, when the first pattern is presented to the net the dendritic sums of the output units will all be multiples of w_0. Marr suggests that feedforward inhibition applied locally to the dendrites of the output units is able to counteract this excitation, and can be set to allow the desired fraction of output units to fire (α_B). The synapses of these output units are then modified; in the model under study in this thesis, the weight is set to unity.

For the storage of subsequent patterns, the inhibition applied must not only counteract the unmodifiable component of the Brindley synapses, but must also increase with the number of modified synapses on the units. It is suggested that feedback inhibitory mechanisms could perform this function, but the exact mechanism is unspecified.

Let us consider the desired properties of the representations constructed. Units should not be active in the representation just because they have higher dendritic sums due to having modified synapses from involvement in the representations of some other patterns. Actually, for the associative memory function, we would prefer units which have been involved in the fewest other patterns. To realize this, other workers have proposed that the units have a conscience: a unit should not fire if it already fires in the representations of a number of other inputs (Grossberg, 1988). Also, given random uncorrelated input patterns, the representations should be random and uncorrelated as well. The question is whether a mechanism that exploits input activity can construct such representations.

A mechanism based on mismatch detection (Gray, 1982) is proposed here. Basically, the normal mode of operation of the net is deemed to be recall mode. If an unlearned input pattern is presented to the net, each output unit will take a dendritic sum from the distribution for the low units. This will be noted by the threshold setting mechanisms - in 'Guess s' the probability of spurious firing will be very high in order to get the desired number of output units firing. In this case, the net can switch to 'learn mode'. In learn mode, the firing of the output units is simply a k-winner-take-all threshold on the activity vector. If no

threshold can be found that will give exactly (or very nearly) M_B active units, the net uses the smallest threshold on the activity vector which yields greater than M_B active units. Those units that fired just at that threshold are noted, and based on their unit usage history which has been recorded (either explicitly or implicitly by modifying the unit), those which have fired least are chosen to be in the representation of this input pattern. From a mechanistic point of view, all of the information required to do this is available.

Simulations were performed using the canonical network parameter set in order to examine the output representations formed by this mechanism. 1000 patterns from one random pattern set were presented for storage. If the output representations are random, the distribution of the number of active bits that overlap between pairs of patterns is very nearly approximated by a binomial distribution $b(N, \alpha^2)$. Figure 5.1 shows the frequency of the overlap between output representations formed after storage of 1000 patterns and $b(1024, (30/1024)^2)$. The binomial models the overlap well. However, this is not the true distribution of the overlap since each output vector has exactly M_B active units. The value of the overlap is generally 0 or 1 bit; the expected number of active bits which overlap is .88.

Thus, this mechanism maps input vectors which have low pairwise overlap to representations that have low pairwise overlap, which is what is desired.

The first question of this chapter was whether appropriate output representations could be developed by the net using some mechanism which exploits the number of active input fibres impinging on an output unit. Appropriate in this case means that random input patterns should map to random target representations. A mechanism has been developed here that meets this specification.

Once a set of input patterns have been stored in the net and output representations for them created, the question is what kind of recall performance can be expected when partial or noisy cues are presented to the net. As in Chapter 3, we begin by studying the distributions of the dendritic sums of the low and genuine output units.

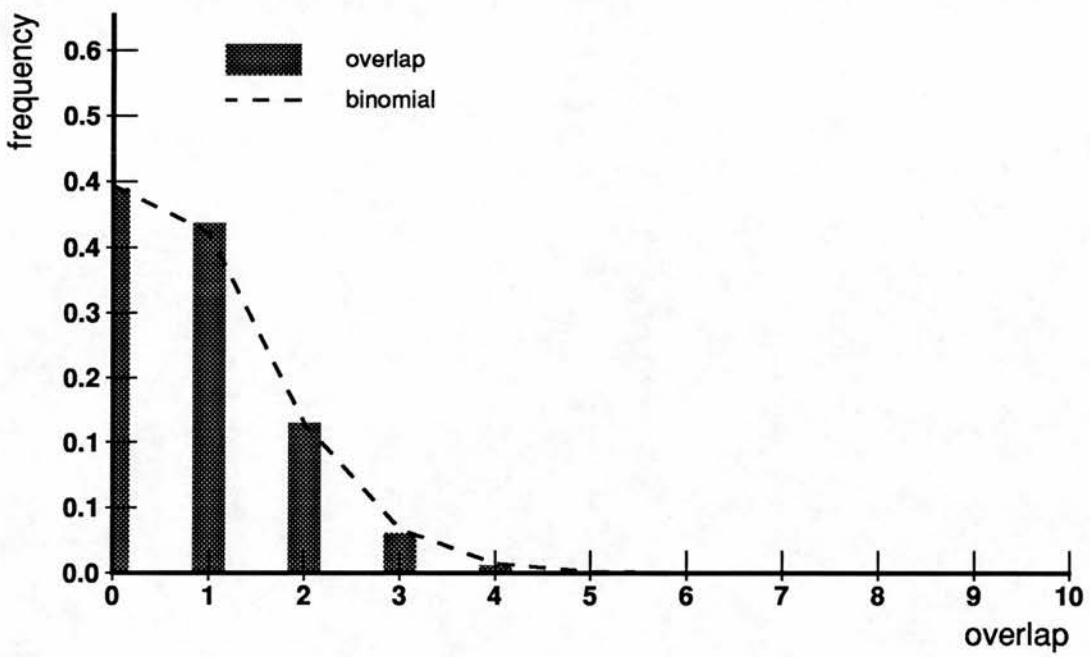

Figure 5.1: Overlap of representations developed by the self-organizing mechanism together with $b(1024, \alpha_B^2)$ which predicts it.

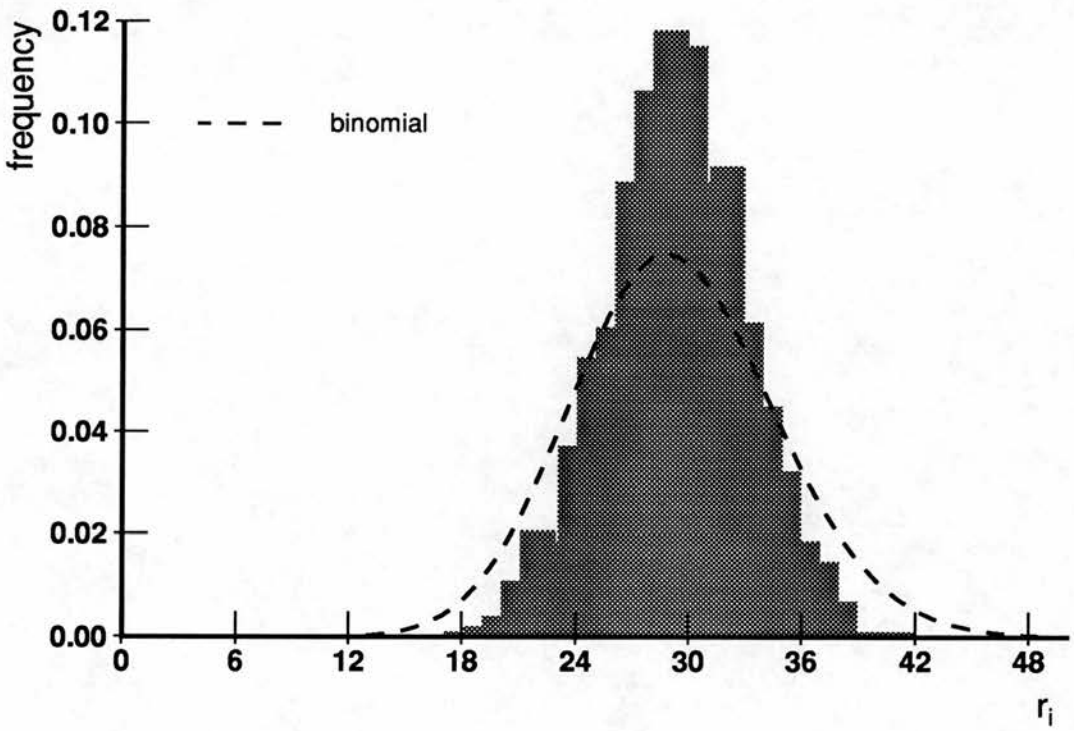

Figure 5.2: Unit usage of representations developed by the self-organizing mechanism together with $b(R, \alpha_B)$ which predicts it. ($R = 1000$)

5.2 Dendritic sum distribution in the self-organizing net

Given a stored input pattern presented to the net, consider the distributions of the dendritic sums for the genuine and low output units. For a low unit, d_s is distributed $b(a_i, \rho_i)$, where the activity a comes from the binomial $b(m_{cue}, Z)$. We need to calculate the value of ρ_i for the self-organized net. Synapses were modified during training when there was conjoint pre- and post- synaptic activity. Units were chosen to be active in the output representation of an input pattern because they had more active afferent synapses than other units for that pattern, so with respect to the active lines in the input pattern, the connection density onto the genuine units is not Z, but is rather higher.

The expressions for the effective connection densities onto the high and onto the low units are now derived. Marr (1971) stated the same results, but did not present an argument to explain them.

Since the activity is distributed $b(M_A, Z)$ during learning and M_B output units are chosen to be active, they are the ones with activity $a \geq t$ where

$$\alpha_B = \frac{M_B}{N_B} = \sum_{x=t}^{M_A} \binom{M_A}{x} Z^x (1-Z)^{M_A - x} \tag{5.1}$$

To calculate the effective connection density onto genuine units, first consider that the probability that any unit will have activity a is given by

$$P(a = x) = \binom{M_A}{x} Z^x (1-Z)^{M_A - x}$$

However, the genuine units are constrained to have activity greater than or equal to t, and the area under the tail of the binomial in equation 5.1 is α_B. So the probability that a genuine unit has activity $a \geq t$ is

$$P(a = x) = \frac{1}{\alpha_B} \binom{M_A}{x} Z^x (1-Z)^{M_A - x}$$

so the expected value of activity for the genuine units is

$$E[a|genuine] = \frac{N_B}{M_B} \sum_{x=t}^{M_A} \binom{M_A}{x} Z^x (1-Z)^{M_A - x} x$$

which is near t. The effective connection density, Z_g, is then

$$Z_g = \frac{E[a|genuine]}{M_A} = \frac{N_B}{M_A M_B} \sum_{x=t}^{M_A} \binom{M_A}{x} Z^x (1-Z)^{M_A - x} x$$

For the canonical example, t = 174 and $Z_g = .753$, which is greater than the value $Z = .666$ that holds in the associative case.

This higher effective connection density onto the genuine units also implies that the connection density onto the low units effectively somewhat lower than Z. The expression for the effective connection density onto the low units is now derived. Let $Z_g = Z + z_g$. The amount of 'extra' connectivity devoted to the genuine units in the self-organized case is then $M_B z_g$. This is effectively taken away from the low units since the connection density across all the units is Z. There are $N_B - M_B$ low units and we assume that this loss is shared out among them equally. So if we denote the effective connection density onto the low units by Z_s, it is given by

$$Z_s = Z - \frac{M_B z_g}{N_B - M_B} = \frac{N_B Z - M_B Z_g}{N_B - M_B}$$

So in the canonical example, $Z_s = .663$. Marr noted both of these cases, but this was one of the areas where his explanation is found to be lacking.

Since the connection density is effectively higher onto the genuine units during training, more synapses receive active inputs, so the fraction of them modified is higher than it was in the associative case. The connection density Z does not appear in the expression for $\rho(k)$, so how is this to be taken into account? The effectively higher connection density makes α_A look higher to the output units. Let α'_A denote the input pattern activity ratio as seen by the output units. It is given simply by

$$\alpha'_A = \frac{Z_g}{Z} \alpha_A$$

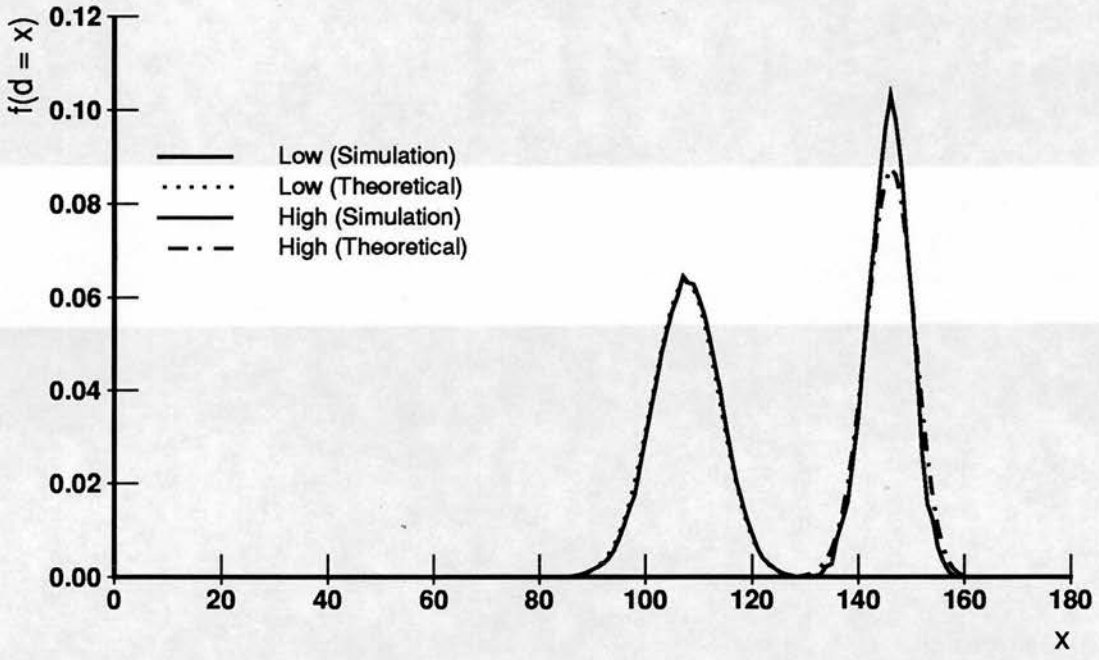

Figure 5.3: Dendritic sums for fixed activity: simulation frequency compared with theoretical distributions. The data are from a simulation run of a net defined by the canonical parameter values, $R = 1000$. Data are plotted for cues such that $s = .4$ ($m_g = 144$, $m_s = 96$), and $a = 170$, $r_i = 30$.

In the self-organizing associative net, the fraction of modified synapses on an output unit involved in k patterns is then given by

$$\rho(k) = 1 - (1 - \alpha'_A)^{k-1} \tag{5.2}$$

Thus in the case of the example, $\alpha'_A = .0339$ and the fraction of modified synapses on a unit which fires in 30 patterns is $\rho(30) = .64485$, which is much higher than that in the associative case.

With these modifications, when the trained net is presented with a (possibly noisy) cue, the dendritic sum distributions for the low and high units are given by the same binomials as those presented in Chapter 3: for the low units, the d_s are distributed $b(a_i, \rho_i)$ and for the high units, the d_g are distributed $b(a_i, 1 - s(1 - \alpha'_A)^{r_i})$.

As in section 4.2 (page 69), the simulation frequencies for the genuine units have a slightly lower variance than the theoretical distribution used.

Using the canonical network parameters, simulations were performed to collect data on the dendritic sum distributions; the canonical parameter values were used to construct the net and 1000 patterns were stored. Dendritic sum frequencies observed in simulations and the theoretical distributions are shown in Figure 5.3 for $a = 170$ and $\tau_i = 30$; the match is very good. One key point affecting recall performance in the self-organizing case is that the distributions overlap more than they do in the straight associative case due to the higher fraction of modified synapses. However, the higher overlap of the pure binomial distributions need not translate into higher expected output errors. Figure 5.4 shows the scattergram of (a, d) observed in simulations for R = 1000, $\tau_i = 30$. Compared to the Figure 4.7 in Chapter 4, we see that the clouds of points are consistently higher than those for the associative case, but in addition, the clouds are offset from each other with respect to activity. The cloud depicting the genuine units is to the right of that of the low units. In the associative case it was directly above the one for the low units. This is due to the differing activity reaching the low and genuine units. In the associative case, the activity on all units came from the same distribution, $b(m_{cue}, Z)$, but in the self-organizing case, the activities onto the genuine and low units come from different distributions. The genuine units are much more likely to have high activity than the low units.

To summarize the chapter so far, we have seen that the partially connected net can self-organize output representations of random input patterns presented to it. We have also seen that the dendritic sums on the units can be described by simple binomials on the activity impinging on the units, though the binomials are somewhat different than those which obtain in the straight associative case. We can now evaluate the performance of the self-organizing partially connected net in the content-addressable memory task.

5.3 Recall performance using Guess s

Recall performance of the self-organizing partially connected net was investigated using the Guess s thresholding strategy. However, in this case, in order to minimize the expected output errors, the *probability* that a low or high unit

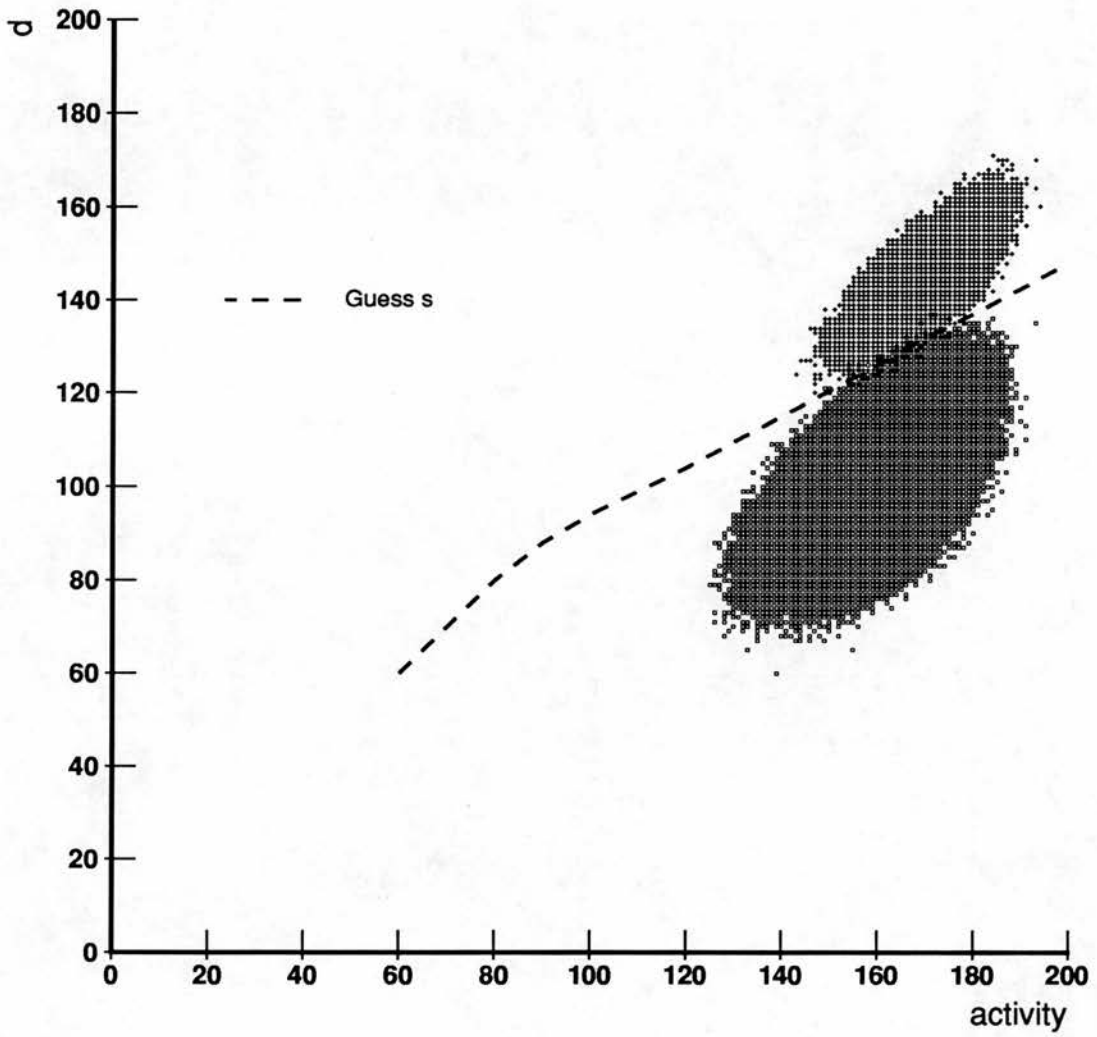

Figure 5.4: Scattergram of dendritic sums and activity observed in simulations together with the Guess s thresholds for the self-organizing net. The (a, d) data come from a simulation run in which the canonical parameter values were used, R = 1000. Data are shown for units with $\tau_i = 30$ and for noisy cues such that $s = .4$ ($m_g = 144$, $m_s = 96$).

will have a particular input activity must be taken into consideration. At higher values of input activity, it is more likely to be a genuine unit so the threshold can be set lower than it would have been in the associative case and still give a low expected number of false positives. Conversely, a unit with low input activity is very unlikely to be a genuine unit, so the threshold can be set higher. Thus the threshold 'line' has lower slope and higher intercept than the one in the associative case. The thresholds that the version of Guess s for the self-organizing net selects for cues with the fraction of spurious bits in the cue, $s = .4$ and $\tau_i = 30$ are shown superimposed upon the clouds of Figure 5.4.

Simulations were performed using the self-organizing training scheme and this 'Guess s' thresholding strategy during recall. Canonical network parameters were used. Figure 5.5 shows recall errors as a function of cue hamming distance for a run in which 1000 patterns were stored together with the expected output error. Comparing these results to those shown in Chapter 3 for a net with 1000 pattern pairs stored does not show much difference. At first, one might think that since the self-organized net chose the output units that have the most active afferents to be active in a representation, on recall the thresholding mechanism should have little difficulty determining those genuine units. However, is should be noted that the fraction of modified synapses in the self-organized net is much higher than in the associative one, which has an adverse effect on performance.

To illustrate how performance changes as more patterns are stored, Figure 5.6 shows results of simulations in which $R = 200, 400, \ldots, 2000$. As in the associative case, recall errors increase with loading, but performance is good over a wide range of R.

5.4 Comparison with competitive learning

The self-organizing partially connected net is an example of a competitive net in that the output units compete (implicitly) to be involved in the representations of input patterns. One of the motivations for the study of this net is that it has been suggested that regions of the hippocampal formation are standard competitive

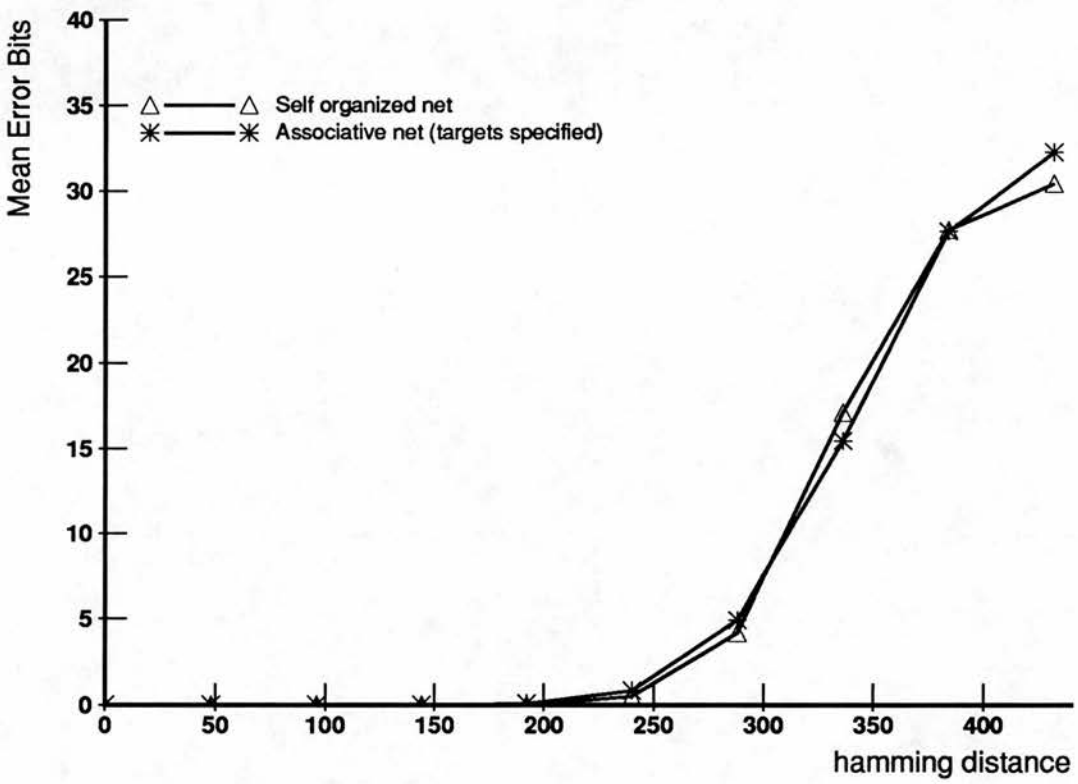

Figure 5.5: Recall performance of the self-organized net as a function of the hamming distance between the cue and the input pattern compared with that of the net in which the targets were specified *a priori* of Chapter 3. Noisy cues were used, R = 1000.

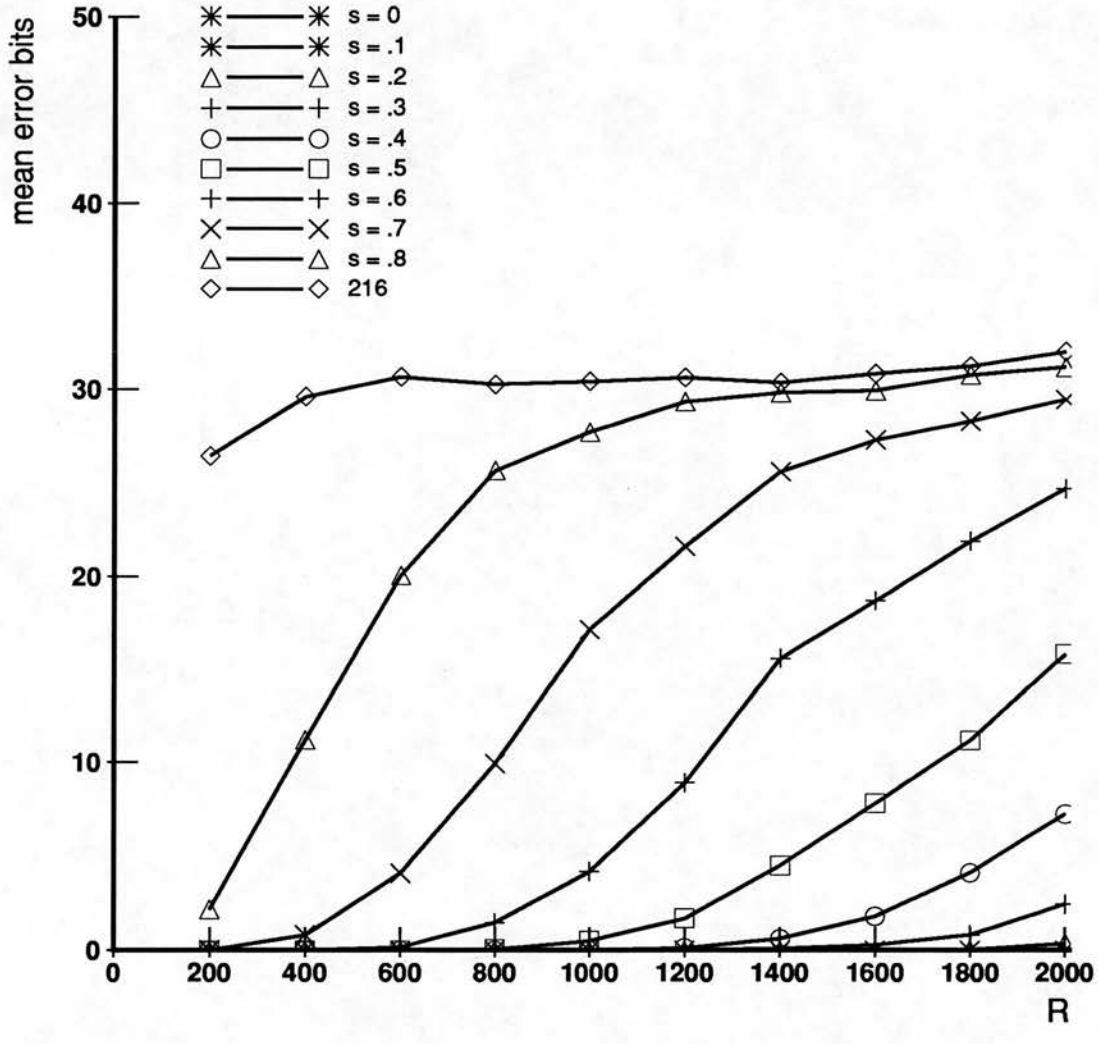

Figure 5.6: Recall performance of the self-organized net as a function of the number of patterns stored in the net. Simulations were performed with the canonical parameter values. Noisy cues were used for recall (with $\delta_g = m_s$).

nets that function as pattern completion devices (Rolls, 1989). This section compares and contrasts the self-organizing net with a simple competitive net (Rumelhart & Zipser, 1985) with respect to the content-addressable memory task using partial cues.

The standard competitive net is described briefly here. Many variations are possible. A full treatment of competitive nets is beyond the scope of this thesis, but a comparison of the standard formulation with the net developed in this thesis is useful.

The standard competitive net has two sets of units, input and output, connected by directed weighted links. The units take a state, which may be real-valued or binary. The weights are real-valued. When an input pattern is presented to the net, the input state vector is multiplied by the weight matrix to yield a dendritic sum vector. A winner-take-all operation is performed on this vector to determine which output unit fires. Most of the work on these nets has concentrated on the winner-take-all case, but in other work multiple output units are active. Thus the output units 'compete' to be active.

To construct one of these nets, the weights are initialized to random values, and then normalized so the the sum of the weights in each weight vector is a constant (usually 1.0). To train the net, patterns are selected at random from the set of input patterns to be learned. Each pattern is presented to the net, the dendritic sum vector obtained, and a winner-take-all operation performed to produce an output vector. Then the weights are modified. Weights are only changed on the unit(s) that fires. The weights that are receiving active input are increased by some small amount and the others decreased. Thus, in subsequent presentations of that pattern, the output unit which has previously won the competition and fired is more likely to win again since its dendritic sum will be even higher than than it was initially.

The weight change expression used in the standard formulation is

$$\Delta w_{ij} = \begin{cases} 0 & \text{if unit } i \text{ does not win} \\ \epsilon \frac{a_{ij}}{a_i} - \epsilon w_{ij} & \text{if unit } i \text{ wins} \end{cases} \tag{5.3}$$

where ϵ is some small positive number and is the rate at which the weights

change. In this expression, a_{ij} denotes whether weight j on unit i is receiving active input: it takes value 1 if so, 0 otherwise. One property of this weight change rule is that the sum of the weights in a weight vector remains constant.

The entire pattern set is presented to the net many times. Presenting all the patterns is called one training cycle or epoch. After a number of cycles, the weight vector of a unit comes to reflect the average of the input vectors for which the unit fires. If the unit only fires for one pattern, its weight vector comes to point in the same direction as that of the input vector (the angle between them is 0). This class of net generally maps similar input vectors to the same output vector – it is a clustering mechanism.

The partially connected net and the standard competitive net are similar in that:

- Both self-organize their output vectors for a given input vector.
- Both can be designed so that a (relatively) fixed number of output units are active in each representation.
- Both use random initial weight configurations to break symmetry.

However, there are important differences, namely:

- Weights in the competitive net are real valued while those in the partially connected net are binary.
- The random initial element in the partially connected net is *where* the synapses are placed, not their initial weight (that is, what subset of the input units synapse with an output unit). In the competitive net, the random element is the initial value of the weight.
- Training in the partially connected net is one-shot while many cycles can be required to change the weights significantly in the competitive net.
- The competitive learning algorithm is designed to keep the summed weight of an output unit's weight vector constant. In the partially connected net, the number of synapses a unit has remains constant, but the

fraction modified varies as more patterns are learned, and more important, varies from unit to unit.

The standard competitive net has been put forward as a device for implementing a content-addressable memory (Rolls, 1989). However, most research on this architecture focuses on its performance in pattern classification tasks and not on its behaviour as a memory device. Still, let us compare the performance of this architecture with that of the partially connected net in the content-addressable memory task.

Simulation results were used to make the comparison. The standard competitive net was implemented so that simulations could be performed using the the canonical parameter set. This is a partially connected competitive net. The locations of the weights are chosen randomly, and each is assigned a small random positive number such that the sum of the weights in a weight vector is 1.0.

As for the self-organizing net, the first question is whether the net can map random input patterns to random output vectors. In general, competitive nets cluster similar input patterns to the same output pattern, but for the memory task, each input pattern must map to a unique output pattern. If the learning rate is high, an output unit which fires for one input pattern will quickly tend to win for other input patterns that overlap in even a few bits with it. So in order to maximize the chance that the competitive net will construct unique output representations for each input pattern, the learning rate, ϵ, should be set quite low. However, the input patterns considered here are random with low activity ratio so none of them is very similar to any other, making clustering less likely.

Simulations of the competitive net were performed using networks defined by the canonical parameter values in which 1000 patterns were stored using various learning rates. An M_B-winner-take-all operation was performed during training. Due to the random initial configuration of weights, in the first training cycle each random input pattern maps to a random output representation. The goal is to modify the weights slowly enough so that the output representations remain random. A number of training cycles were performed, usually 50. It

was found that the algorithm produced random output representations (with respect to pairwise overlap and unit usage measures) when $\epsilon \leq .01$ was used for the learning rate. This indicates that the competitive net can form the unique representations required by the memory task.

The next question is whether the weights have been modified enough to ensure robust recall from partial cues. For any one output unit, before learning the weights are random. During learning the weights which received active input when the output unit fired will have been increased and the others decreased. The weight vectors were studied after training and indeed, after approximately 40 training cycles, they consisted of either high weights (weights with value near $\frac{1}{S\rho(k)}$) or low weights (values near zero).

Let us consider how different the weight vectors of different output units are before and after learning. Since the weights are real-valued and the sum of the weights is a constant, the cosine of the angle between two weight vectors provides a reasonable measure of their similarity. Given weight vectors initialized as described above with 8000 elements and 5333 nonzero weights, the expected value of a nonzero weight is 1/5333 and the cosine of the angle between two such vectors is simply $Z = .666$. From simulations, the average cosine of the angle between two weight vectors before training is .62. During the training process, the expected proportion of the weights that have been increasing is $\hat{\rho}$ as before. After training, these weights will share out most of the weight among them, while the value of the others will tend to 0. Thus the expected value of a weight after training is $1/(S\hat{\rho})$ and the expected value of the cosine of the angle between two weight vectors is $Z\hat{\rho} = .39$ (assuming no structure in the input pattern set). The mean value obtained in simulations after training is .31. So, after training the competitive net the difference between weight vectors is greater than before training.

Next, simulations were run in which a number of patterns were stored, learning (weight modification) turned off, and recall performance tested. Simulations were run for $R = 200, 400, \ldots, 2000$.

Results as mean output errors are shown in Figure 5.8 for partial cues as a function of the number of patterns stored; each line in the graph depicts results

obtained with one value of the number of missing bits in the cue. Compare this with the results of the same experiment for the partially connected net constructed with the same parameters using the Guess s thresholding strategy (Figure 5.7). The recall behaviour is quite different. For a given cue, as the number of patterns stored increases, the mean errors increase nearly linearly for the competitive net while for the partially connected net they remain low and finally increase steeply. The difference is not surprising since the two use very different thresholding strategies.

Over most of the range of R, the performance of the partially connected net is much better than that of the standard competitive net. If the task is good recall using a partial cue with ten percent of the genuine bits active, the self-organizing net can store approximately 1000 patterns while the competitive net can store approximately 200.

Comparing information efficiency: 32 bits are used to represent each weight in the competitive net and its performance is only 20 percent of the performance of the partially connected net (in this task), thus its information efficiency is $\frac{1}{80}$ of that of the partially connected net.

5.5 Conclusions

Mechanisms can be devised which enable the partially connected net to construct its own output representations for input patterns presented to it. The self-organizing net performs well in the associative and content-addressable memory tasks, although for a given number of stored patterns the representation construction scheme causes a higher fraction of synapses to be modified than in the associative case, which has an adverse effect on performance. However, it performs significantly better than the standard competitive net. Thus the self-organizing partially connected net, motivated by Marr's theory of the archicortex, is a viable architecture for self-organizing content-addressable memories.

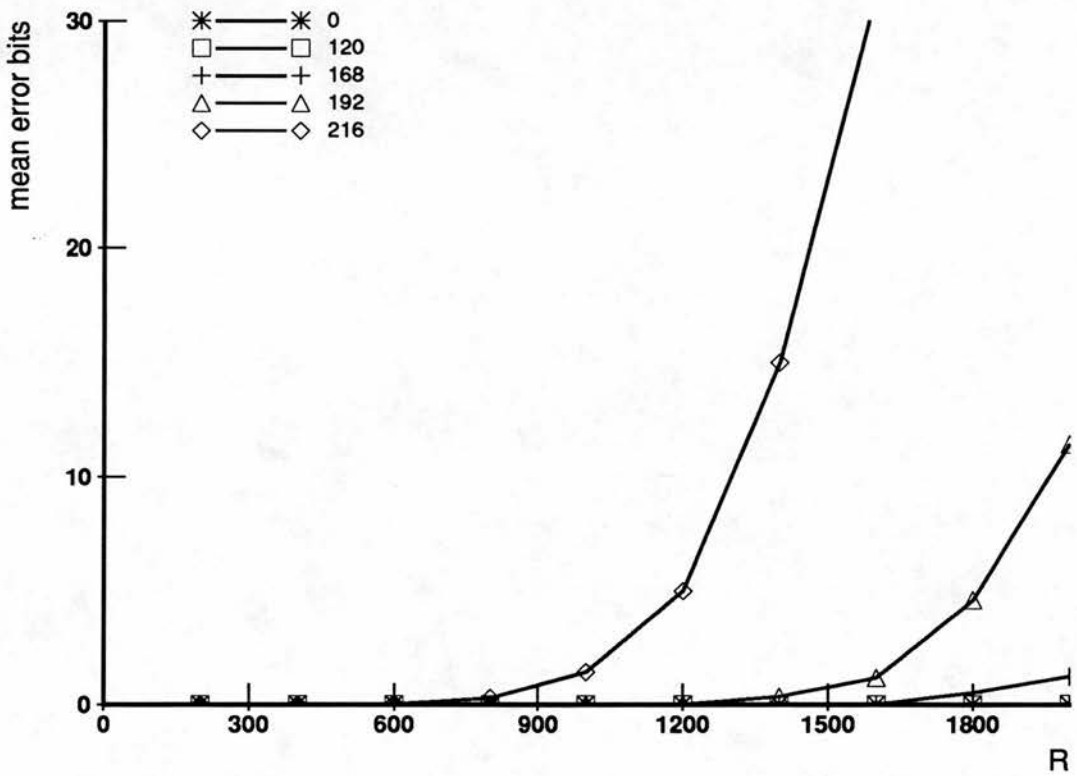

Figure 5.7: Mean output error of the self-organizing net as a function of the number of patterns stored, R. Simulation results are shown for a number of partial cues; each cue, denoted by cue hamming distance (in this case the number of missing bits), has one line in the graph.

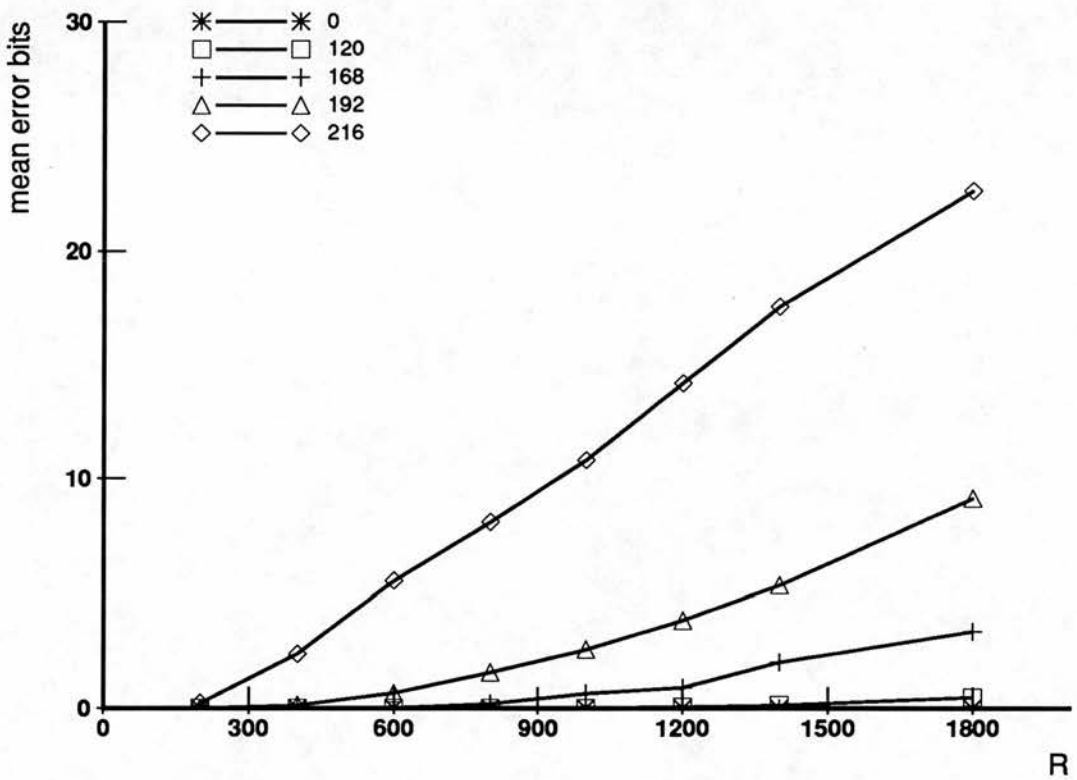

Figure 5.8: Mean output error of the standard competitive net as a function of the number of patterns stored, R. Simulation results are shown for a number of partial cues; each cue, denoted by cue hamming distance (in this case the number of missing bits), has one line in the graph.

Chapter 6

Applications of Activity Based Thresholding

This thesis has shown that partially connected networks with binary valued synapses can indeed function as associative memories. We have explored performance issues with respect to various thresholding techniques, the number of patterns stored, and network parameter settings. This chapter reviews the results developed in the previous chapters and discusses their relevance to other network models of associative memory and to understanding the functioning the hippocampus. Use is made of numerical analysis rather than simulation since most of the nets studied in this chapter are too large to be simulated

6.1 Capabilities and limitations of the partially connected associative net

Partially connected associative nets can perform the associative and content-addressable memory tasks well. The primary factors which affect performance are the fraction of synapses on an output unit that are modified during the training process and the number of active input lines which make contact with an output unit. The lower the fraction of modified synapses, the better the recall. The larger the number of active input lines that reach a unit, the better

the recall. The exact effects that both of these have can be simply derived from the overlap of the dendritic sum distributions for the desired active and inactive units.

The optimal parameters for a network depend on the task that it should perform. There is a tradeoff between the number of patterns which can be stored with good recall and the content-addressable functionality of the net. If the simple associative memory task is required, many more patterns can be stored than if recall from partial or noisy cues is required. Viewed another way, if content-addressablity is not required, the net can be made much more sparse than if it were required. This is because the amount of activity reaching the output units must be large enough that the dendritic sum distributions for the low and the high units can be effectively distinguished. As noted by other workers and as shown by the results of Chapter 4, the expected activity of the genuine units, MZ, should be greater than 20 for good recall.

Thus these nets should really be large nets. M must be large to keep $m_{cue}Z \geq 20$ and if M is large, N must be large to keep α, and hence ρ low. In large nets obeying these constraints, the connection density can be low and the net will still deliver good recall performance. Delicate nets, vivid recollection.

6.2 Several applications of Guess s

The understanding we have gained about the factors underlying the behaviour of sparsely connected associative nets should help us to understand related models better. This section revisits progressive recall and then discusses multi-layer associative nets.

6.2.1 Progressive recall

Progressive recall was one of the first thresholding strategies developed for a sparsely connected autoassociative net (Gardner-Medwin, 1976). Using pro-

gressive recall, the net can function as a content-addressable memory. However, the method only really works for partial recall cues. Spurious bits in the cue tend to recruit other spurious bits. Its performance could be improved if activity dependent threshold strategies were employed.

Conversely, the performance of an autoassociative partially connected net using the Guess s strategy can be improved by exploiting progressive recall. Using Guess s as described in Chapter 4, if a partial cue is clamped onto the units, all of the genuine units will be recruited. However, some number of spurious units may also fire. In some cases, the number of spurious units elicited by a partial cue is high. But if only a few new units were recruited, the ones with the highest dendritic sums, the number of spurious units could be reduced.

Let us consider an example. In Chapter 3, the performance of progressive recall was plotted as a function of the number of patterns stored for a net parameterized by N = 8000, M = 240 and Z = .25 using partial cues with 48 genuine bits active (Figure 3.12). Using this net, two new sets of simulations were performed. In the first, Guess s alone was used for recall. In the second, the 'm++' progressive recall method using Guess s for thresholding was used. Simulation results for 'm++' (from Chapter 3), Guess s alone, and 'm++ with Guess s' are shown in Figure 6.1; mean output error is plotted as a function of R for partial cues with 48 genuine bits active. We see that Guess s on its own performs better than progressive recall on its own in this example. But more importantly, progressive recall using Guess s performs much better than either on its own.

As implemented, this only holds for partial cues since the method stops cycling when the desired number of units are active. Given a noisy cue with $m_{cue} = M$ in an autoassociative net, it may be possible to reduce the final output error by cycling the net using Guess s, but a comprehensive treatment of the dynamics of autoassociative nets is beyond the scope of this thesis.

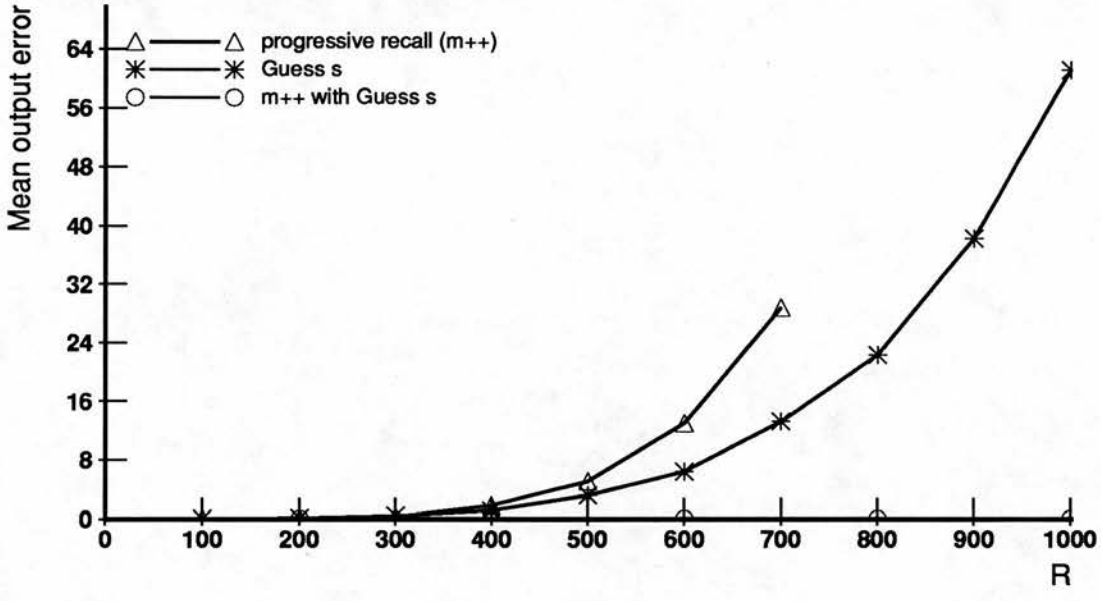

Figure 6.1: Comparing the mean output error of progressive recall using Guess s to set unit thresholds with progressive recall on its own and Guess s on its own. For the simulations, an autoassociative net defined by N = 8000, M = 240 and Z = .25 was used; results are shown for partial cues with 48 genuine bits active. The progressive recall method used was m++.

6.2.2 On multi-layer associative network architectures

More complex architectures than those considered in Chapters 3, 4 and 5 can be constructed by linking several nets. A comprehensive treatment of multi-layer associative network architectures is beyond the scope of this thesis, but the tools required to make statements about them are supplied by the design constraints and analysis of storage capacity of individual nets developed here.

There are a number of reasons to consider multi-layer associative nets. They may be the appropriate model for the system under study. Marr used one because he felt that a three layer structure modelled the anatomy of the hippocampal formation.

Willshaw (personal communication) has suggested that they might be used for a type of 'impedance matching'. It may be the case that the pattern pairs to be associated together have parameter values which do not suit the associative net. In some cases the situation can be improved by inserting another layer of units between the input and output units. For instance, if both the input and target patterns have relatively high activity ratios, then the associative net would not perform effectively as a memory device. However, the main reason that this is the case is that high activity levels leads to high ρ. This can be counteracted by having a middle or hidden layer of units in which α is very low. If the input and output layers are called A and B as usual and the hidden layer called H, in this architecture the A$\rightarrow$H net would be a self-organizing net and the H$\rightarrow$B net an associative one. Layer H is to be constructed so that α is very low, but $M_H Z_B$ must still be at least 20 so the number of units in layer H may end up being large. Small α_H will help keep ρ_H and ρ_B low, which will enable the structure to function as an associative memory.

An example will serve to show how knowledge of the performance characteristics of single nets can be used in the design of a multi-layer one. A memory task is specified, then a network architecture designed that will perform up to the specification.

The task:

- Storage of 10,000 pairs of vectors with binary elements. The input and target vectors are specified by $N_A = N_B = 10000$, and $M_A = M_B = 1000$.
- Good content-addressable recall from a partial cue with 10 percent of the genuine bits, with good recall defined by the expected number of output errors being less than or equal to 1.
- Some tolerance to noise in the cues is required.

A fully connected associative net would not be able perform this task since the maximum number of pattern pairs that can be stored and still obtain good recall in a net of this size is approximately 350. However, a multi-layer structure can be designed that can. A layer of units (called H for hidden) is inserted between the input and output layer. The input units (layer A) project only to units in layer H, which project to the output units (layer B). The A $\rightarrow$H net is self-organizing, and the H $\rightarrow$B net is an associative net.

In the design of this architecture, the parameters under our control are N_H and α_H, the size and activity level of the hidden layer respectively, and Z_H and Z_B, the connection density between layers A and H and layers H and B respectively. To find values for these parameters such that the resulting structure can perform the task specified we need to consider the fraction of synapses modified during training and the activity reaching the layer H and layer B units.

The fraction of modified synapses in the layers is the most important attribute to control. Following the results of Chapter 4, we want $\hat{\rho} \leq .6$ for both weight matrices. Since $R = 10000$ and $\alpha_A = \alpha_B = .1$, α_H is set to 0.001.

Now we set the number of active units in layer H. For good recall of the layer B pattern, the expected activity of the layer B units during recall should be high. (It should be at least 20, but above 50 makes it easier to distinguish the dendritic sum distributions of the low and high units).

$$M_H Z_B > 50$$

implies that M_H must be at least 50. For this example, let us use partially connected nets as has been done throughout the thesis. Since M and Z can both

vary in order to satisfy the activity constraint, let us simply fix one of them. Let $Z = .6$. Then $M_H > 84$ so let $M_H = 100$ simply to overdesign the structure. This gives $N_H = 100,000$.

Similarly, for good recall of the layer H representation from a partial cue presented to layer A,

$$m_{cue} Z_H > 50$$

gives $Z_H > .5$. Again to overdesign the structure, let $Z_H = .6$. The parameter values that define this architecture are set out in the following table:

	A	H	B
N	10,000	100,000	10,000
M	1,000	100	1,000
α	.1	.001	.1
Z		.6	.6
S		6000	60000

Using the Guess s threshold strategy in these partially connected nets, the expected numbers of false positives and negatives given the number of genuine and spurious bits in a cue can be calculated using the expressions developed in Chapter 4. To meet the task specification, a cue with 100 genuine bits and no spurious bits must elicit good recall in the output layer. After storing 10,000 pattern pairs, the expected numbers of genuine and of spurious units in the layer H representation elicted by this cue are calculated to be 100 and 3 respectively. Assuming that the expected values are obtained, this representation will elicit 999.99 genuine and 0.018 spurious units in the output layer (expected values). Reducing the connection densities to 0.5 in both nets gives expected values of 100 genuine and 24 spurious units in layer H, and 996 genuine and 4 spurious units in layer B.

The number of connections in this structure is an order of magnitude more than that for a fully connected network from A to B, but then, the fully connected network is not able to store 10,000 pattern pairs. Other architectures are possible. In this task, the hidden layer is required because the level of activity of the input and target patterns is so high. The connection density of the A $\rightarrow$H and

H →B nets could be made much lower if greater numbers of false positives and negatives on layer B were acceptable. Or to improve recall performance, a set of collateral connections between the layer H units could be installed and the layer H representation cycled before projecting it to layer B.

6.3 Relevance to understanding the hippocampus

The mammalian hippocampus is crucially involved in the establishment of episodic memories. Its architecture resembles that of associative network memories and so many have suggested that it functions as one. If one of its functions is that of an associative memory, it would be useful to clearly lay out the constraints on its operation. Important work contributing to an understanding of such network memories has been done by Willshaw *et al.*, Marr and Gardner-Medwin. Willshaw and colleagues characterized the functionality of the fully connected associative net. Marr (1971) employed sparsely connected associative nets in his theory of the archicortex. Gardner-Medwin (1976) analyzed progressive recall in a sparsely connected autoassociative net motivated by the collateral projection in the CA3 region of the hippocampal formation. The work presented in this thesis collects and extends their analyses.

This section briefly notes the functions that the hippocampus is involved in and presents a sketch of the anatomy of the structure. An improved estimate of the recall performance of Marr's model of the hippocampus is derived using the work developed in this thesis. Finally, a speculative estimate of the storage capacity of the rat hippocampus is calculated.

6.3.1 What the hippocampus is involved in

Most of the evidence on what the hippocampus is involved in comes from neuropsychological studies in which the ability of human patients or animal subjects with hippocampal damage to perform some task is compared with that of normal subjects. Subjects with hippocampal damage (who will be referred

to as hippocampals) display deficits in a wide range of tasks, with memory impairments being the most pervasive.

The hippocampus has been implicated at various times in processing related to olfaction, emotions and memory, but its involvement in memory is by far the most extensively studied. In 1957, a patient known as H.M. had a bilateral temporal lobectomy to treat intractable epilepsy which resulted in complete anterograde amnesia with respect to learning new facts or episodes (Scoville & Milner, 1957). The severity of the deficit motivated a number of research efforts to investigate the role of the hippocampal formation and other temporal lobe structures in memory. All mammals studied exhibit deficits in memory function after hippocampal lesions, though the nature of the deficit varies. For example, in humans the deficit covers mostly memories for facts and events, while in rodents deficits in the ability to perform tasks which require spatial navigation are most evident. O'Keefe (1971) found cells in the rat hippocampus whose firing is correlated with the physical location of the test animal in a familiar environment which led him to postulate that it is the site of spatial memory in rodents. Olton (1976) found that hippocampal animals are impaired in tasks that require remembering items relevant to the task at hand and suggested that it is the site of 'working memory'. The ability to learn new 'procedures', like learning to ski, is usually not impaired. Also, hippocampal damage does not result in sensory deficits. Hippocampal animals can still perform simple sensory discrimination tasks (Morris, 1983).

In humans, the hippocampus appears to be required in order to learn new facts or episodes, but not for retrieval of already learned ones. It has been suggested that it is involved in the 'consolidation' of declarative memories, but not in their long-term storage (Marr, 1971; Squire *et al.*, 1989; Zola-Morgan & Squire, 1990). In rodents, the situation is not as clear. Morris *et al.* (1982) report that rats that are well trained to find a hidden platform in the water-maze task are unable to find that location after hippocampal lesions. It is not clear how network memories could support a spatial memory.

6.3.2 Anatomical sketch of the hippocampal formation

The hippocampal formation lies below the mantel of the cerebral cortex and on top of the midbrain. It consists of a set of interconnected populations of nerve cells which form the edge of the neocortical sheet: entorhinal cortex, subiculum, the hippocampus proper and another sheet of cells called the dentate gyrus.

The hippocampus itself is shaped like two bananas hanging downward connected in the middle. In gross terms, each of these banana shaped lobes is actually a sheet rolled up like a swiss roll. Early anatomists gave various names to this structure. Some thought the 'S' shape of the rolled sheet of cells looked like a seahorse, or in Latin, *hippocampus*. Others thought the lobes looked like rams horns and called the structure 'Horns of Ammon' or *Cornu Ammonis*. They further subdivided it based on the anatomy of the cell types and nerve fibre connections and imaginatively named the regions CA1, CA2, CA3 and CA4 (Lorente de No, 1934).

The hippocampal formation is interconnected with various cortical and subcortical brain structures. It is reciprocally connected to cortical structures primarily via the entorhinal cortex, while the main subcortical pathway is via the fornix to the septum and the mammillary bodies. In general, the pathways between structures are made up of the axons of primary cells, usually large pyramidal cells. There are many other kinds of neurons presents in these structures; in general their axons project only within their structure so are often called interneurons; many are thought to exert an inhibitory influence on the primary cells. Figures 6.2 and 6.3 summarize the major excitatory cortico-hippocampal pathways. For more extensive reviews see Rosene and Van Hoesen (1987) and Eichenbaum *et al.* (1990).

6.3.3 Theories of hippocampal function

There are a number of theories regarding the function of the hippocampal formation, and most of them require that the hippocampus is able to function as a memory device (O'Keefe, 1976; McNaughton, 1989; Olton & Samuelson,

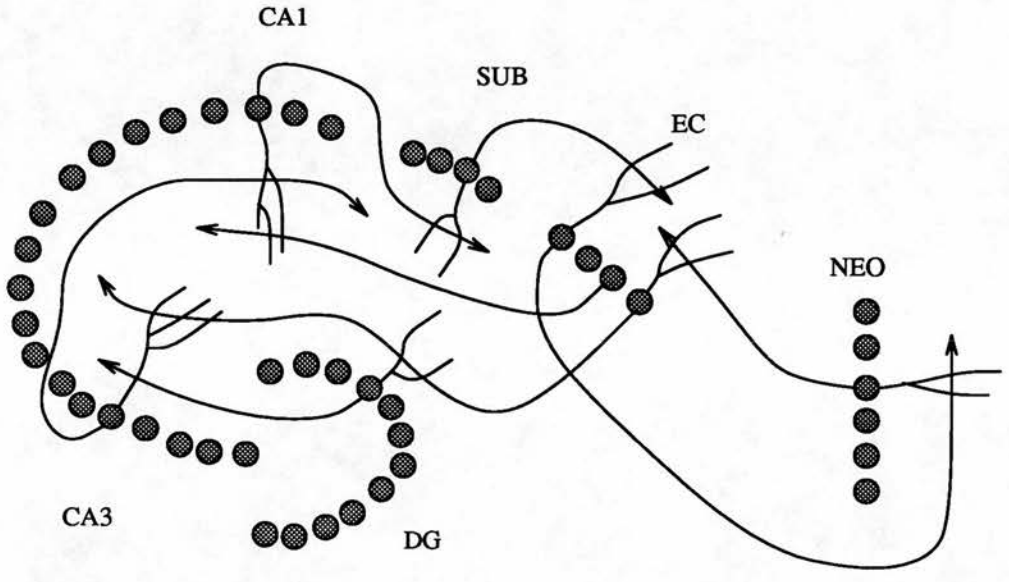

Figure 6.2: Drawing of the major excitatory cortico-hippocampal pathways (Squire, Shimamura and Amaral, 1989). The filled circles represent the primary cells (either pyramidal cells or granule cells). Neocortical pyramidal cells (NEO) project (via the perirhinal cortex) to layer II and III entorhinal cortex cells (EC). The axons of the layer II EC cells form the perforant path which penetrates the hippocampal fissure and projects to dentate gyrus (DG) granule cells and CA3 pyramidal cells. EC layer III cells project directly to CA1 pyramidals. The DG granule cell axons, the mossy fibres, make strong excitatory synapses onto the dendrites of the CA3 pyramidals. The axons of the CA3 pyramidals split into three pathways: (i) collateral projections synapse widely onto other CA3 cells, (ii) one pathway leaves the hippocampal formation via the fornix, and (iii) a final pathway (shown here), the Schaffer collaterals, projects to CA1. The CA1 pyramidal cells project primarily to the subiculum (SUB) although some project directly back to EC. The subiculum (and CA1) project to the deep cells of EC, which project back to the neocortical areas which originally projected to EC, thus completing the loop.

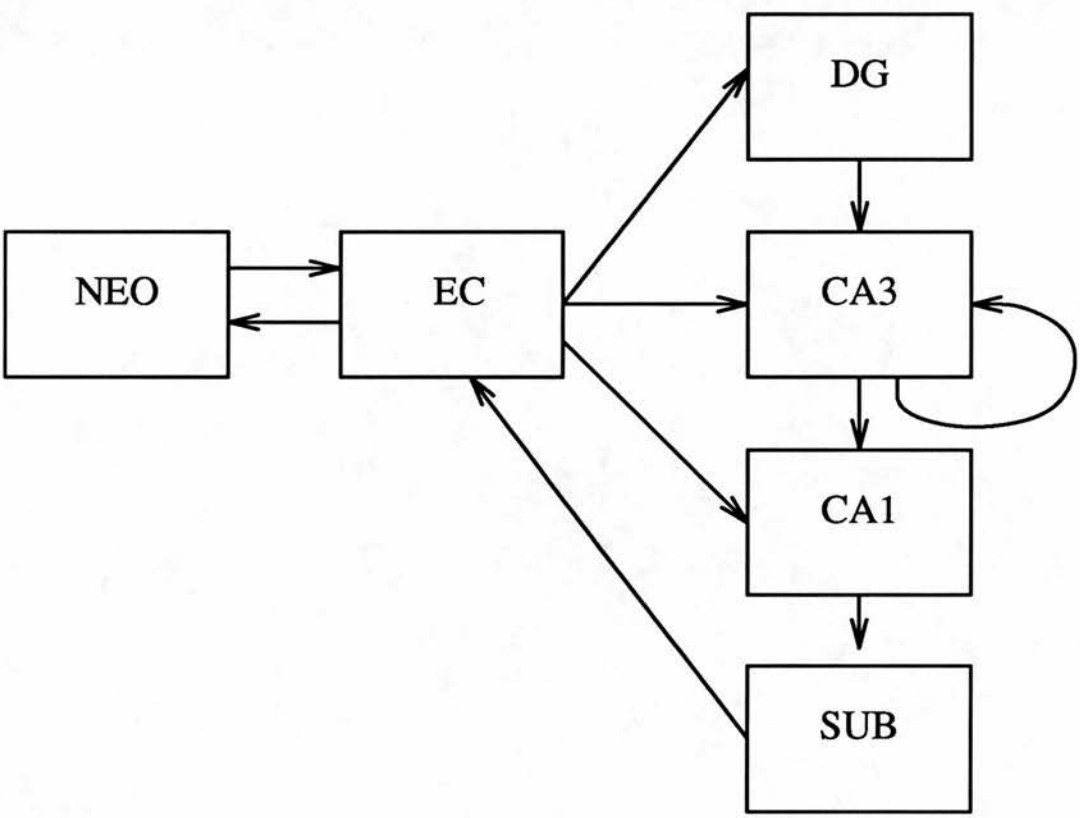

Figure 6.3: Schematic of the major cortico-hippocampal pathways.

1976; Rawlins, 1985; Gray, 1982; Eichenbaum *et al.*, 1988; Rolls, 1989). Several of these theories assume that parts of it act as network memories (Marr, 1971; McNaughton, 1989; Gray, 1982; Rolls, 1989). Most of these theories are based on evidence from neuropsychological studies and are concerned with characterizing the deficits observed in hippocampal subjects. In addition, others are based on neurophysiological evidence regarding the observed activity of neurons in the hippocampus. With the exception of Marr's work, these are not formal theories. The hypotheses in them that parts of the hippocampal formation implement network memories make appeals to the anatomy. They do not compare experimental findings to the analyses of network models. Though there have been suggestions that the hippocampal formation functions as a network memory, there is currently little experimental evidence to support this idea.

To date, Marr's theory of the archicortex provides the most rigorous account of how the hippocampus might work. Since this theory motivated much of this thesis, it is discussed in light of the work presented here.

6.3.4 Marr's model of the hippocampus

Marr (1971) proposed that the mammalian hippocampus acts as a temporary content-addressable memory store. He first discusses why a structure specialised for temporary storage is needed at all. In his related theory of the neocortex (Marr, 1970), he proposed that the primary function of the principal cells of the neocortex, the *pyramidal* cells, is the reorganisation and classification of information. However, neocortical connectivity is not extensive enough to support all the associations between two pyramidal cells which may be required. In addition, there is the need to store new information as it arises. In the hippocampus paper, Marr argues that it would be inefficient to store such transient information in the permanent memory of the neocortex until it is known what features of the new information are required. So a temporary memory structure is required to both store this information and provide the structural link between physically disparate neocortical areas.

The goal is content-addressable recall. A number of *events*, each represented

as a pattern of activity in a selected population of nerve cells, is to be stored. Subsequent presentation of a small part of a previously stored event must then enable the whole of it to be reconstructed. The model he develops consists of three populations of cells, $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ (Figure 6.4). $\mathcal{P}_1$ is the input layer and corresponds to the neocortical input to the hippocampal formation, $\mathcal{P}_2$ models entorhinal cortex, and $\mathcal{P}_3$ the CA regions of the hippocampus. $\mathcal{P}_3$ is the output of the model.

Events are presented to $\mathcal{P}_1$ and representations are self-organized on $\mathcal{P}_2$ as described in Chapter 5. The projection from $\mathcal{P}_1$ to $\mathcal{P}_2$ is divided into 25 blocks to model the spatial organization of the projection from neocortical areas onto entorhinal cortex. A representation of the $\mathcal{P}_2$ pattern of activity is then formed on layer $\mathcal{P}_3$. Collateral connections between the $\mathcal{P}_3$ units model the collateral projection in the CA3 region of the hippocampus. Thus $\mathcal{P}_1 \rightarrow \mathcal{P}_2$ and $\mathcal{P}_2 \rightarrow \mathcal{P}_3$ are self-organizing nets and $\mathcal{P}_3 \rightarrow \mathcal{P}_3$ is an associative net.

The parameter values that define the model are set out in Table 6.1. The task that Marr sets for this structure is content-addressable recall using a partial cue with 10% of the genuine units active after 100,000 events have been stored. Marr's main tools of investigation were mathematical analysis and numerical solution of the equations he formulated for the various computations envisaged. He concludes that this model can perform the task specified.

Willshaw and Buckingham (1990) assess Marr's theory. In this paper, an explanation of the derivation of the parameter values of this model is presented, Using Marr's assumptions and expressions for the dendritic sum distributions, the expected number of genuine and spurious units are recalculated and found to match those stated by Marr. They find that the theory has several innovative features such as the activity based T,f thresholds discussed in Chapter 3, but several considerations lessen its appeal.

1. It is a poor model of the hippocampal formation. Marr's choice of the structure of the model seems to have been influenced heavily by his view that the hippocampal formation is a three-layer network, a view constructed somewhat independently of his computational results. Although he reviews in some detail the cells that are to form the various proposed layers, only a

Overall parameters of $\mathcal{P}_1 \rightarrow \mathcal{P}_2 \rightarrow \mathcal{P}_3$:

	1	2	3
N	1,250,000	500,000	100,000
M	2,500	3,025	200
α	0.002	0.006	0.002

The block structure of $\mathcal{P}_1 \rightarrow \mathcal{P}_2$:

	1	2
N	50,000	20,000
M	100	121
α	0.002	0.006
S		10,000
Z		0.2

$\mathcal{P}_2 \rightarrow \mathcal{P}_3$ and the $\mathcal{P}_3$ collaterals:

	2	3	3′
N	500,000	100,000	100,000
M	3,025	200	200
S		50,000	10,000
Z		0.1	0.1

Table 6.1: The parameters which define the architecture of Marr's model of the hippocampal formation. The parameters for the $\mathcal{P}_3$ collateral projection are denoted by 3′.

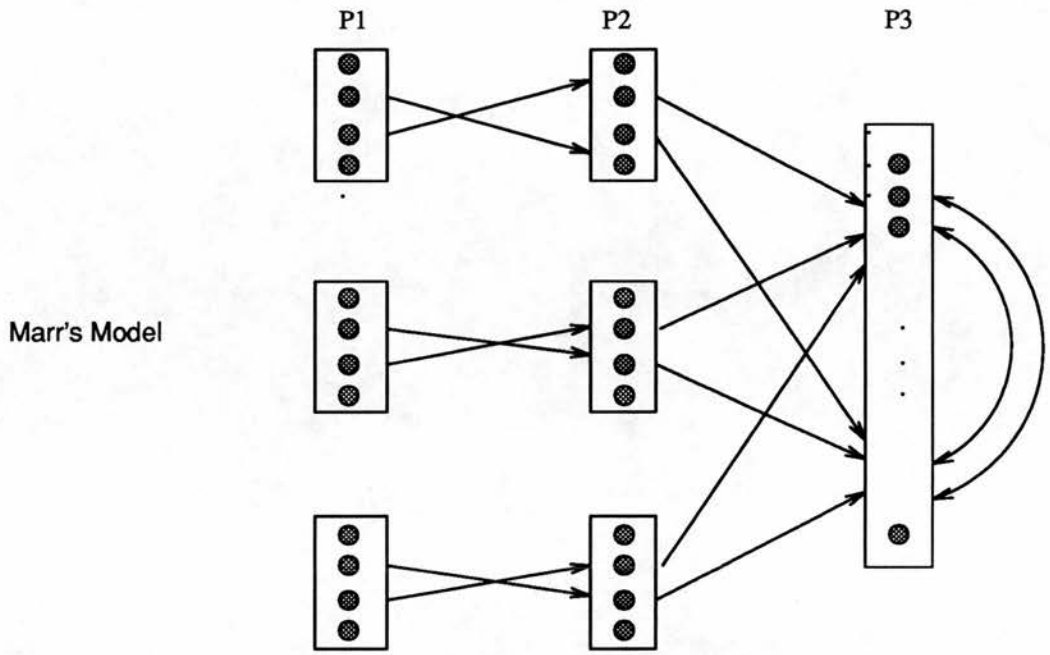

Figure 6.4: Architecture of Marr's models. $\mathcal{P}_1$ corresponds to the neocortical input to the structure, $\mathcal{P}_2$ to the entorhinal cortex, and $\mathcal{P}_3$ to the CA regions of the hippocampus.

loose correspondence between the sub-divisions of the hippocampus and the layers of the model is made. The most extensive discussion revolves around the nature of layer $\mathcal{P}_3$. In identifying the memory elements of this layer with the CA pyramidal cells of the hippocampus, he is placing less importance on the Dentate Gyrus-CA3-CA1 trisynaptic circuit (Figure 6.2) (Andersen *et al.*, 1971) than might have been expected considering the emphasis on that circuit at the time.

In these respects, Marr presents a somewhat abstract interpretation of the hippocampus as a temporary memory – in sharp contrast to his theory of the cerebellum (Marr, 1969). A thresholding mechanism that can exploit activity information is a key feature of his model, but he does not provide an account of how the purported inhibitory cells in the hippocampal regions could support this function. Perhaps his most important contribution was that he provided many detailed predictions, such as those concerning the level of activity and the way synapses are modified during learning.

The model requires that synapses are modified by simultaneous pre-synaptic and post-synaptic activity. It pre-dates the finding of Long Term Potentiation (Bliss & Lømo, 1973; Bliss & Gardner-Medwin, 1973) in the hippocampus, although he does add a note in proof about Lømo's earlier paper (1971) showing synaptic facilitation in the perforant path - dentate gyrus pathway. Unfortunately, most of his predictions have not been followed up.

2. No convincing computational arguments are given for deciding that the temporary memory required by the neocortex must be a three layer net. In this paper on the hippocampus he alternates between two claims: (1) that the structure of the simple memory proposed must necessarily be so for it to act as a content-addressable memory; (2) that it has to have this structure because the hippocampus is built like this. Marr recognized this himself and in later work (1982) discusses the shortcomings of this sort of modelling.

3. The analysis of the net is sometimes unsatisfying. The expressions he develops for the probabilities of false positives and negatives obscure understanding the behaviour of the network. They have several other properties that lessen their appeal.
(i) There are thresholding strategies that exploit input activity information that provide better recall performance than the dual T,f thresholds suggested by Marr and which do not require the omniscience assumed in Marr's paper.
(ii) In common with analyses of the capacity of associative networks (with the exception of Willshaw and Dayan (1990)), the analysis which gives rise to these expressions does not take into account that output units fire different numbers of times, so the values calculated using these expressions are not in accord with the actual behaviour of the net.

A much simpler analysis from which come accurate, computationally simpler expressions for the expected numbers of genuine and spurious units firing in one of these nets given the number of genuine and spurious bits in the cue is presented in Chapter 4.

Despite these shortcomings, Marr's intuitions about the behaviour of sparsely connected nets are excellent. His model functions very well as a content-addressable memory.

Improved estimate of recall performance of Marr's model

Since Marr did not consider unit usage in his analysis, the performance predictions are somewhat high. In this section, the storage capacity of his model is recalculated using the expressions for the dendritic sum distributions developed in Chapters 4 and 5.

The expected numbers of genuine and spurious bits in the patterns elicited in layers $\mathcal{P}_2$ and $\mathcal{P}_3$ can be calculated using Guess s. Those expected values are calculated here for the cues which Marr considered (see Table 6.1 for the parameters that define this architecture).

In this model, the $\mathcal{P}_1 \rightarrow \mathcal{P}_2$ network blocks and $\mathcal{P}_2 \rightarrow \mathcal{P}_3$ are self-organizing nets and the $\mathcal{P}_3 \rightarrow \mathcal{P}_3$ collaterals form an associative net. For a given cue presented to layer $\mathcal{P}_1$ defined by a number of genuine and spurious bits, the expected numbers of genuine and spurious bits in the pattern elicited in $\mathcal{P}_2$ are calculated. Using these expected values to define the $\mathcal{P}_2$ cue, the expected numbers of genuine and spurious bits on layer $\mathcal{P}_3$ are calculated and so forth.

Marr considers the case of recall cues in which cells in just one of the $\mathcal{P}_1$ blocks are activated. The work presented in the previous two chapters makes it clear why, from a computational viewpoint, it is wise to divide the $\mathcal{P}_1 \rightarrow \mathcal{P}_2$ projection into blocks. Consider recall from a cue with 100 genuine bits active and no spurious bits imposed on one of the $\mathcal{P}_1$ blocks. With the connection density he employs for $\mathcal{P}_1 \rightarrow \mathcal{P}_2$, $Z_2 = .2$, this gives an expected activity of 20 on the genuine units of $\mathcal{P}_2$, which is about as small as it can go for reasonable recall to be possible. Without any block structure, this would imply that the number of synapses on each $\mathcal{P}_2$ cell would be 250,000, which is quite large. This is large by computational standards, since synapses require storage and during recall computation may have to be performed using each synapse, and also by biological standards – most pyramidal cells do not have that many synapses on their dendrites. However, by dividing up the projection into 25 blocks, the number goes down to 10,000, which is much more reasonable.

In the next few tables, the expected number of genuine and spurious bits which

arise when cues defined by m_g and m_s are presented to $\mathcal{P}_1$ are shown for structures in which 100,000 events have been stored. As in Marr (1971), let B_0 and B_1 denote the number of genuine and spurious bits in the pattern of activity on $\mathcal{P}_2$ and C_0 and C_1 those on $\mathcal{P}_3$.

In Marr's model, the $\mathcal{P}_1 \rightarrow \mathcal{P}_2$ projection is divided into 25 blocks. The blocks do not interact so for this projection it suffices to examine the recall performance of one block. The number of genuine bits per block in a stored event is 100.

Cue		$\mathcal{P}_2$	
m_g	m_s	E(# genuine)	E(# spurious)
100	0	107	14

Given this $\mathcal{P}_2$ pattern imposed on one of the blocks, the expected numbers of genuine and spurious units in the $\mathcal{P}_3$ pattern are:

$\mathcal{P}_2$ Cue		$\mathcal{P}_3$	
B_0	B_1	E(# genuine)	E(# spurious)
107	14	0	0

Thus, recall from a partial cue composed of all of the genuine units of one block of $\mathcal{P}_1$ is not possible. Using the expressions developed in Chapters 4 and 5, which take unit usage into account, recall is worse than predicted by Marr. With $R = 100000$, poor recall in the form of no pattern of activity on $\mathcal{P}_2$ is the result. The $\mathcal{P}_1 \rightarrow \mathcal{P}_2$ layer is simply overloaded. The structure would perform better if the blocks in this layer were larger and if the connection density were much greater.

The expected values of the numbers of genuine and spurious units in $\mathcal{P}_2$ and $\mathcal{P}_3$ were calculated for a number of values of R in order to determine the number of patterns this structure could store with good recall from a partial cue imposed onto one block of $\mathcal{P}_1$ that has 100 genuine bits active. It was found that this structure can store up to approximately 80,000 patterns before the expected output error exceeds 1. At $R = 80,000$, the expected numbers of genuine and spurious bits on $\mathcal{P}_2$ are 117 and 9 respectively. From $\mathcal{P}_2$ to $\mathcal{P}_3$ the expected numbers are 74 genuine and 65 spurious. Using this as an initial pattern of

activity on $\mathcal{P}_3$, the expected numbers of genuine and spurious units across a number of cycles of the $\mathcal{P}_3 \rightarrow \mathcal{P}_3$ collateral loop are:

Cycle	$E[C_0]$	$E[C_1]$	
0	74	65	
1	57	19	
2	81	75	
3	65	19	
4	102	97	
5	93	20	
6	151	32	
7	189	6	
8	200	0	$\surd$

Note that the expected numbers of genuine and spurious units do not change smoothly at this loading. If the structure is loaded more heavily, recall fails. However, this too is an overestimate of the performance of this structure since the expected values are not always attained, as noted in the discussion of progressive recall in Chapter 3.

6.3.5 Relevance to the real hippocampus

The functioning of the real hippocampus is much more complex than that of a set of linked partially connected nets. Real neurons are much more complex than the threshold units of the partially connected net, even considering sophisticated inhibitory mechanisms implicit in the Guess s strategy. There are many temporal issues to consider in the functioning of real neurons - their activity is not locked into simple recall cycles. Though some hints about the firing correlates of cells in regions afferent to the hippocampal formation are provided by unit recording studies, the nature of the signals which reach it is still very unclear. Brainstem areas that project to the hippocampal formation may be involved in setting the general state of the hippocampus, since some of these areas are purported to play roles in motivation and attention. For instance, dentate gyrus granule cells are more active when an animal is exploring a new

environment. Perhaps this higher activity level is mediated in part by brain-stem influences. Also, it is known that the hippocampal formation is involved in the processing of temporally extended events – simple layered associative net architectures will not account for any of this functionality.

There are some indications from LTP studies that the way synaptic modification behaves may be one area where the models are probably most similar to real neurons. Some studies suggest that synaptic modification in the hippocampus could be a binary phenomena with decay (Friedlander *et al.*, 1990). Whether the decay is active or passive has not been conclusively determined.

A Capacity estimate

With those provisos, considering that theta rythym might clock hippocampal function in a lock-step fashion, the suggestion the individual hippocampal synapses may be be 'binary valued', and the suggestion that the levels of activity in at least some hippocampal areas is relatively low (Thompson & Best, 1989), as an exercise we ask how many patterns the hippocampus could store if it functioned as a simple layered partially connected net structure.

Consider the architecture depicted in Figure 6.3. Entorhinal cortex projects to dentate gyrus and CA3. The dentate gyrus projects via the mossy fibres to CA3. CA3 has its collateral loop, and projects to CA1, etc. Subcortical afferents are assumed to act as a pacemaker and so are not shown. For this exercise, let us consider the storage capacity of CA3. The goal is to store patterns in the structure such that subsequent presentation of partial or slightly noisy patterns will elicit firing of the learned representation of the pattern on the CA3 units.

Entorhinal cortex is taken as the input layer, and CA3 as an associative net. DG is assumed to be a self-organizing net. The firing rate of DG granule cells is usually very low (Green *et al.*, 1990), but increases during exploratory activity in a new environment so assume that they are firing during a 'learn mode' of the hippocampus. Due to the high strength of the mossy fibre synapses on the CA3 dendrites, assume that the firing of the DG granule cells effectively clamps

the firing pattern on the CA3 units. Since the firing of DG units sets the firing pattern of the CA3 units, they can be viewed as providing the target vectors during training of the CA3 net. Thus for the purposes here, DG acts simply as a random vector generator.

In order to make capacity calculations, some values have to be stated for the parameters of the networks. There are reports in the literature regarding some of the parameters, while others must be inferred. More numbers are available for the Sprague-Dawley rat than for many other species, so those are used for this exercise. Approximately 200,000 entorhinal cortex pyramidal cells project via the perforant path to the dentate gyrus and CA3 (Amaral *et al.*, 1990). Boss *et al.* (1985, 1987) report the number of cells in the dentate gyrus, CA3 and CA1 as 1,000,000, 320,000, and 420,000, respectively. These numbers include all cells in the pyramidal layers of these regions[1], but they state that the fraction of other cell types is very small. Of the synapses onto CA3 pyramidals, approximately 4000 of the afferents are perforant path fibres while approximately 12,000 are CA3 collaterals. There have been no explicit reports of the fraction of pyramidal cells active simultaneously in any of these regions; however, activity in DG is known to be normally very low (), and it can be inferred from Thompson and Best (1989) that α_{CA1} is at most 0.1, but nothing definitive can be said about how much less it might be. The structure of entorhinal cortex, CA3 and CA1 are all quite different, so there is little reason to assume that the fraction of active cells will be the same in each.

How should the model be designed? The experimental findings provide important constraints on the possible values of the parameters of an architecture that should model the hippocampal formation, but the requirements of the formal models studied in this thesis highlight the areas where current knowledge is limited. In these areas, assumptions can be made, guided by the findings that have been reported and by the understanding of the behaviour of the formal models studied in this thesis. The storage capacity is calculated for two sets of assumptions about the connectivity and levels of activity in the hippocampus.

Considering connection density first, let Z_{CA3} and $Z_{CA3'}$ denote the connection

[1]Granule cell layer of DG.

densities of the EC to CA3 projection and CA3 collateral projection respectively. Though there is some spatial ordering to the CA3 collateral projection, most CA3 pyramidals send axon processes throughout the CA3 region and so the connection density can be taken as $Z_{CA3'} = 12000/320000 = .0375$. The spatial ordering of the perforant path projection is a less clear case. Though not as tightly restricted to a lamellar projection as once thought, it is not completely uniform either. If any given entorhinal cortex pyramidal cell contacted any given CA3 pyramidal cell with equal probability, then the connection density could be taken to be $Z_{CA3} = 4000/200000 = .02$, but some spatial ordering does exist. Two assumptions are considered: (1) the EC $\rightarrow$ CA3 projection is organized into two blocks, from the septal end to the temporal end which gives $Z_{CA3} = .04$, and (2) the EC $\rightarrow$ CA3 projection is organized into four blocks ($Z_{CA3} = .08$).

Since small changes in α lead to large differences in the storage capacity of the associative network memories treated in this thesis, assumptions are made about it reluctantly, but some statement has to be made in order to proceed so for the first case, let $\alpha = 0.02$, and in the second let $\alpha = 0.01$. With α this high, the number of patterns that can be stored in the individual nets will be fairly low compared with the number of cells.

To summarize the parameter values, first, the fairly well agreed values of anatomical parameters in EC and CA3 are stated followed by the sets of values assumed in the two cases to be considered.

Experimentally reported parameter values:

	EC	CA3	CA3→CA3
N	200000	160000	320000
S		4000	12000

The parameter values that define the two cases to be considered:
Case 1:

	Per block		
	EC	CA3	CA3→CA3
N	100000	160000	320000
M	2000	3200	6400
Z		.04	.0375
S		4000	12000

Case 2:

	Per block		
	EC	CA3	CA3→CA3
N	50000	80000	320000
M	500	800	3200
Z		.08	.0375
S		4000	12000

There are some striking differences between these parameters and those employed by Marr. The connection density in the rat hippocampus is much lower than that of Marr's model and the activity ratio much higher. In Marr's model, due to the high connection density a relatively low number of units need to be active in an afferent layer so that the activity on the genuine cells of the next layer is high enough for the dendritic sum distributions to be distinguished by the thresholding mechanism. The low connection density in the rat hippocampus means that more cells afferent to a particular structure will have to be firing in order to get recall to work if the structures function as associative network memories as described in this thesis.

For the first set of assumptions, $\alpha_{EC} = .02$ implies each full EC pattern has 4000 active units. With $Z_{CA3} = .04$, the smallest cue that still satisfies $m_{cue} Z_{CA3} \geq 20$ has 500 active units. This is slightly larger that a 10% partial cue, but still a cue small enough to usefully test content-addressable recall. When a recall cue with 500 genuine bits and no spurious bits is presented to one block of EC after 5000 patterns have been stored, the expected numbers of genuine and spurious units on CA3 are 3200 and 11913, respectively.

The expected values across cycles through the CA3 collateral loop for $R = 5000$ are also shown in the top graph of Figure 6.5; recall is successful after 6 cycles.

When more patterns are stored, recall using these cues is not successful. Thus this architecture delivers good recall in the content-addressable memory task using 10% partial cues for up to 5000 stored patterns.

Now turning to the second set of assumptions, with the level of activity in EC being .01, each EC pattern has 2000 active units. With the contact density specified by the anatomy, the smallest partial cue that can be used for recall is one with approximately 250 genuine bits. The expected recall performance calculated for this cue imposed on one of the EC→CA3 blocks predicts 632 genuine and 258 spurious units firing in CA3 after 14000 patterns have been stored.

Taking this initial CA3 pattern and cycling it through the CA3 collateral loop, the expected numbers of genuine and spurious units firing are plotted in the bottom graph of Figure 6.5. This graph shows that recall is successful after 4 cycles. Recall is unsuccessful if the structure is loaded more heavily. Thus, if the goal is content-addressable recall using a partial cue containing approximately 10% of the genuine bits, the CA3 region of the rat hippocampus can store roughly 14,000 patterns under these assumptions.

6.3.6 Predictions and questions about the hippocampus

This thesis was motivated by questions arising from studies of the structure and function of the hippocampus. What does an understanding of the functioning of sparsely connected associative nets offer towards enterprise of developing an explanation of hippocampus works? As expected, this understanding gives rise to more questions than answers. However the questions are different from those often addressed by neuroscientists.

But first, let us note some of the predictions which come from this understanding. If parts of the hippocampal formation function as network memories, then as stated by Willshaw, by Marr and by Gardner-Medwin, the level of activity in these regions must be low.

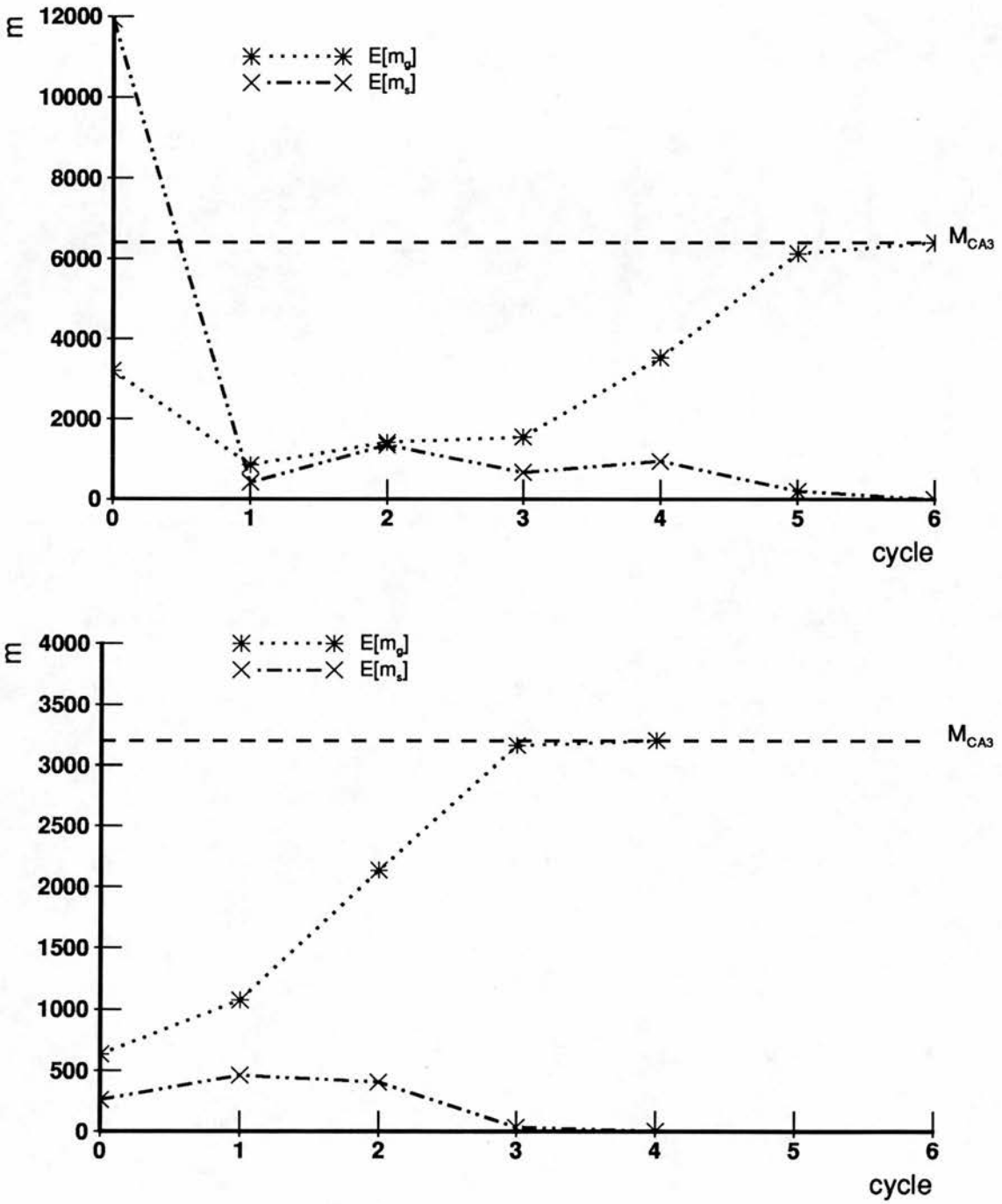

Figure 6.5: Autoassociative recall in CA3. The expected numbers of genuine and spurious units at each recall cycle are plotted. The top graph depicts recall using the first parameter set, the bottom the second.

Marr also stated the idea that the firing thresholds of units should be based on the activity impinging on them. In his theory, activity information would reach the pyramidal units in the form of inhibitory signals proportional to it. The threshold setting strategies developed in this thesis suggest that pyramidal units could do well if their firing threshold machinery were constructed to set thresholds according to the distribution of the dendritic sums on the desired active and quiet units. However, most important, this thesis predicts that if a brain region implements a network memory optimally, then the thresholds must be a function of the firing history of the unit. The bottom line is that thresholds should be set differently for each unit.

These statements suggest questions to be explored in neuroscience. The first question is the most obvious: Does the hippocampus, or parts of it, actually act as a content-addressable network memory? If areas of the hippocampal formation do act as network memories, many more questions arise. How many memories are the areas required to store? What are the values of the parameters relevant to this function, such as the activity levels? There is still little reported about the fraction of pyramidal cells simultaneously active in the hippocampal regions. Experiments need to be performed to determine the activity levels in the hippocampal areas.

Then there are questions related to determining which units are active. Why do the units which fire do so? That is, how does thresholding work? Does thresholding keep a relatively constant number of cells firing in a hippocampal region? If so, does it do so by means of local or more global thresholding, perhaps through inhibitory cells? To set the number of units firing to some desired number might seem to require some more global mechanism, but the input activity measurements needed for the thresholding strategy might best be done locally. Different cell types might mediate the different functions. Basket cells have far reaching axonal and dendritic arborizations, while the spatial reach of some cells in the molecular layer is much more restricted. Do the DG mossy fibres 'clamp' the activity of the CA3 pyramidal units? If CA1 is a self-organizing net, how are the units recruited to be part of the representations of signals impinging on it? What are the temporal characteristics of the structure? Does it operate in a lock-step fashion clocked by a fornix pacemaker signal?

Chapter 7

Conclusions

This thesis has presented a characterization of the storage capacity and recall performance of partially connected associative nets with binary weights. In partially connected nets, one of the key problems is how to set the unit thresholds in such a way that the units which should not fire are quiet and those that should fire do so. New thresholding strategies have been developed which enable this architecture to function well as an associative content-addressable memory. An important result of this work is that in order to do so, the best thresholding strategies are functions of the firing history of an output unit and the number of active inputs impinging on it.

Analysis was presented that predicts the behaviour of the network using the Guess s thresholding strategy. This analysis enables the behaviour of more complex multi-layer structures composed of several partially connected nets to be predicted too.

Further Work

Work which builds on our understanding of the functionality of sparsely connected associative nets can be taken in several directions.

The dynamics of the behaviour of partially connected autoassociative nets need to be studied. The behaviour of progressive recall using partial cues was investigated in Chapter 3, but this is only one case. A number of questions remain, especially concerning the dynamics of their operation.

Given an application requiring an associative or content-addressable memory, the results presented in this thesis provide the necessary design constraints. If the characteristics of the input and target patterns are not amenable to a single net, layered nets can be constructed to do the job. A comprehensive treatment of layered associative networks was beyond the scope of this thesis, but it does provide the foundation for further work on them.

Appendix A

Pattern Sets: Characteristics

This appendix summarizes some of the characteristics of the pattern sets used in the simulations in this thesis.

Each pattern in a pattern set has exactly M elements equal to 1 out of N. The remaining $N - M$ elements are 0. The M elements in state 1, the 'active bits', are chosen randomly. This appendix presents data on two characteristics of the pattern sets: (1) the unit usage, and (2) the pattern overlap.

Given a pattern set containing R patterns, the number of times each unit is active across all patterns, the *unit usage*, is distributed $b(R, \alpha)$ where $\alpha = \frac{M}{N}$. The actual frequency of unit usage in the pattern sets is plotted together with the appropriate binomial that describes it in the figures to follow.

Define pattern *overlap* between patterns x^j and x^k to be

$$\sum_{i=1}^{N} x_i^j x_i^k$$

that is, the dot product[1] of the two patterns. The overlap between two random patterns which have M active bits out of N is distributed $b(N, \alpha^2)$. The actual overlap between pairs of patterns and this distribution are also plotted together in the figures.

[1]This is only a useful measure of overlap for vectors with binary valued elements.

A number of sets of random patterns were generated for use in the simulations for this thesis. Most have N = 8000 and M = 240, or N = 1024 and M = 30. Approximately 20 pattern sets for each were generated. Data for three of each will be presented in the following figures. Figures A.1, A.2, and A.3 show data for pattern sets with N = 8000, M = 240. Figures A.4, A.5, and A.6 show data for pattern sets with N = 8000, M = 240.

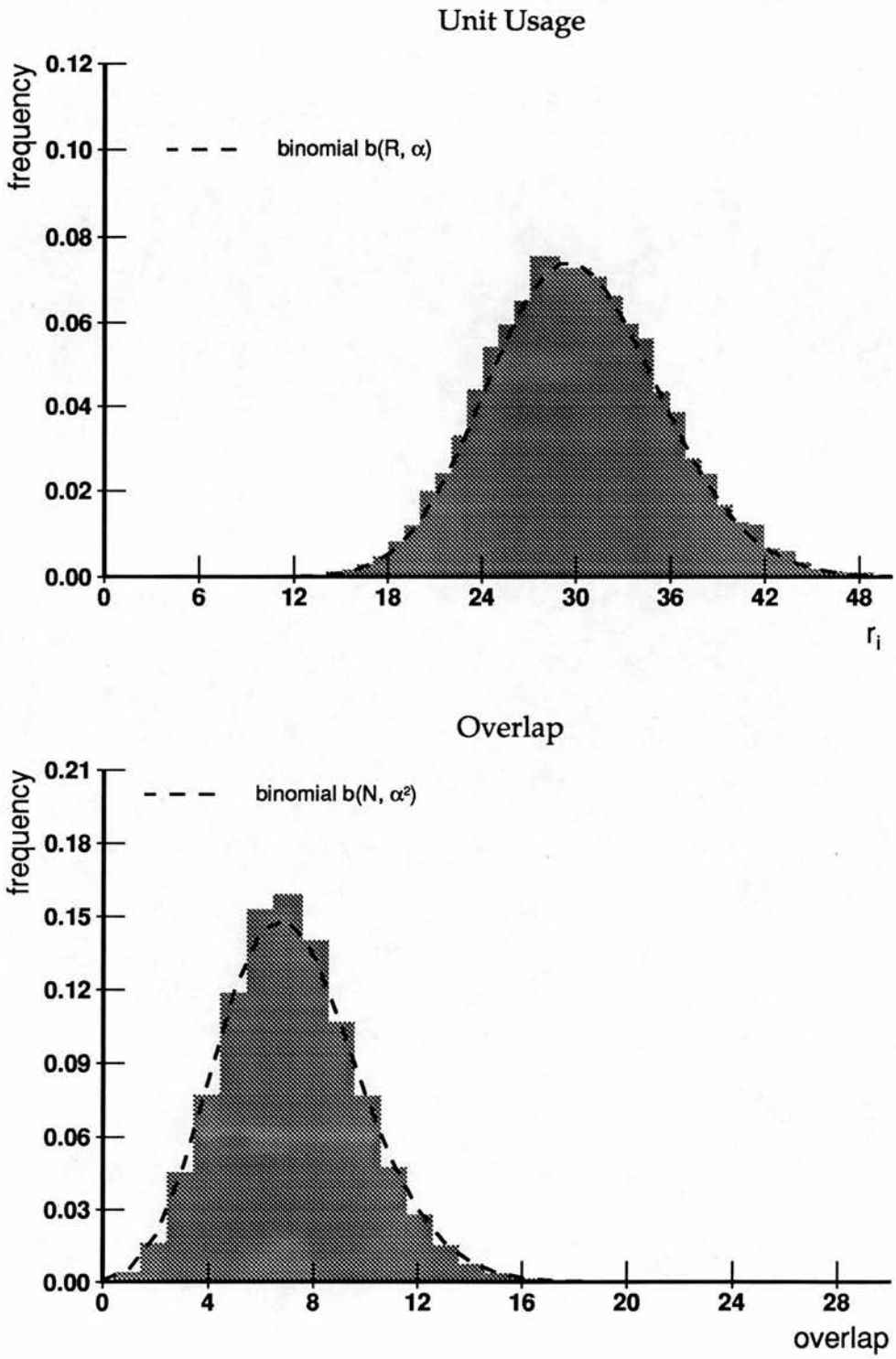

Figure A.1: Data for 1000 patterns from pattern set P2000.N8000.M240.1.input. N = 8000, M = 240.

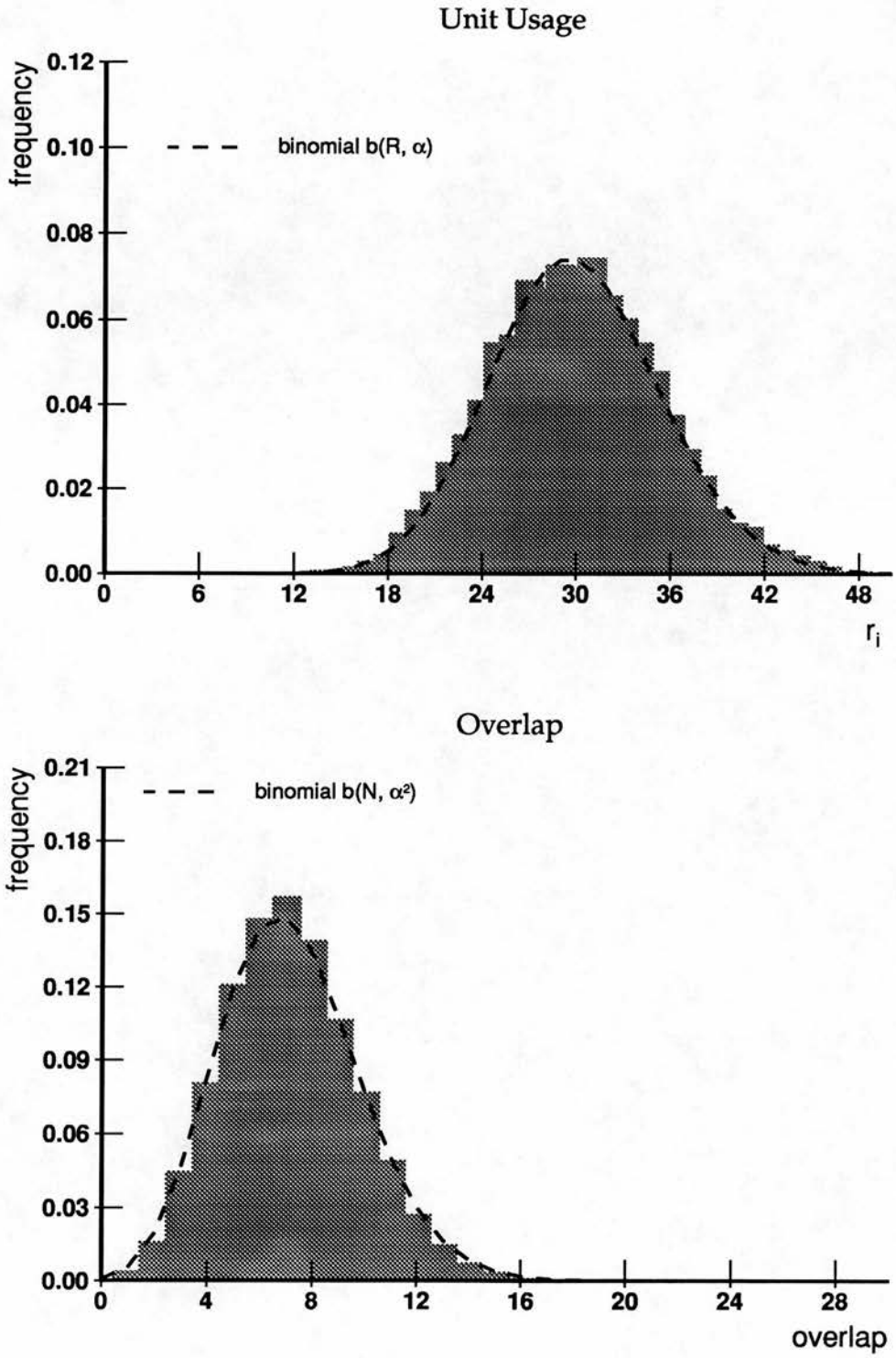

Figure A.2: Data for 1000 patterns from pattern set P2000.N8000.M240.5.input. N = 8000, M = 240.

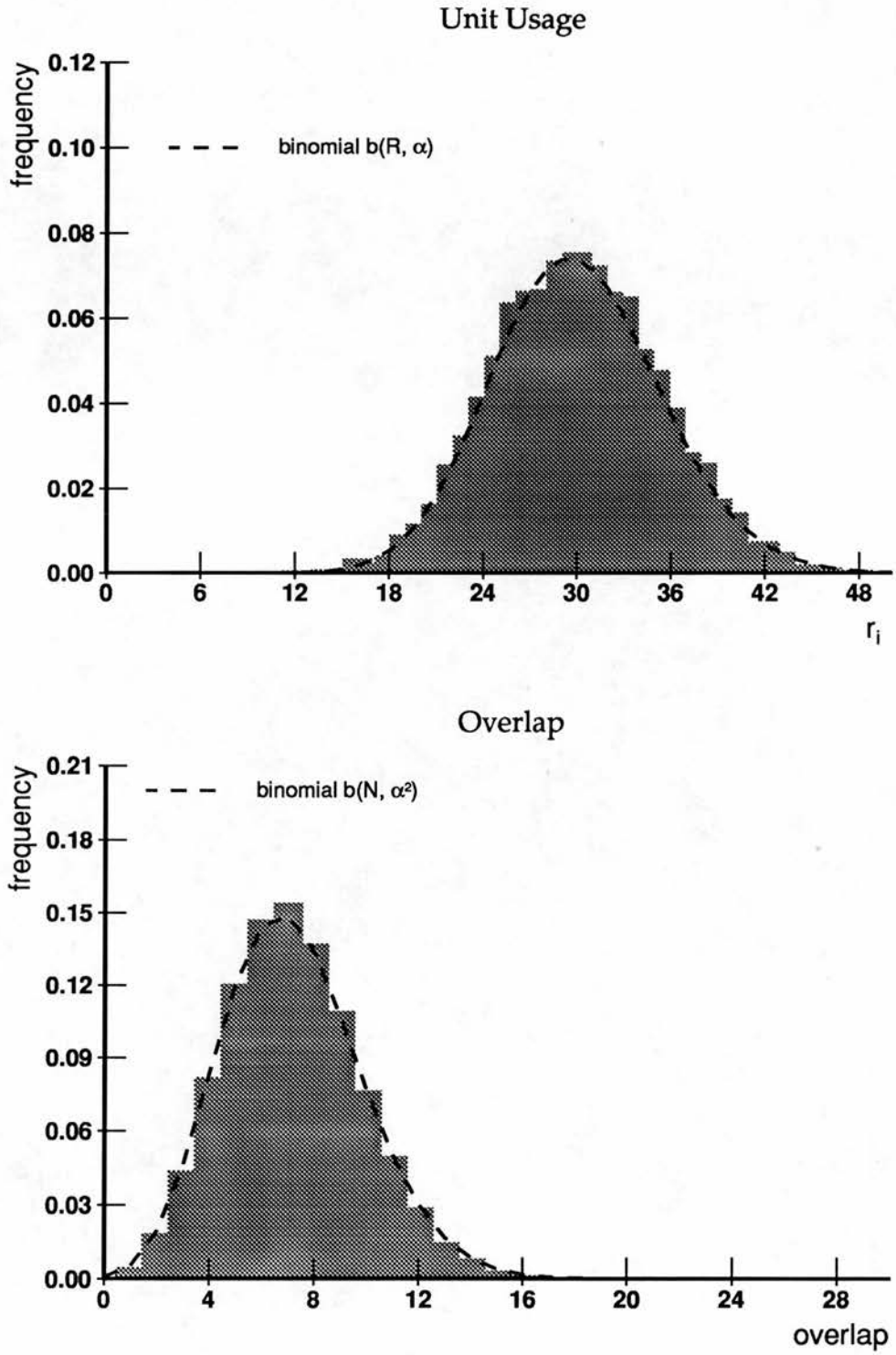

Figure A.3: Data for 1000 patterns from pattern set P2000.N8000.M240.8.input. N = 8000, M = 240.

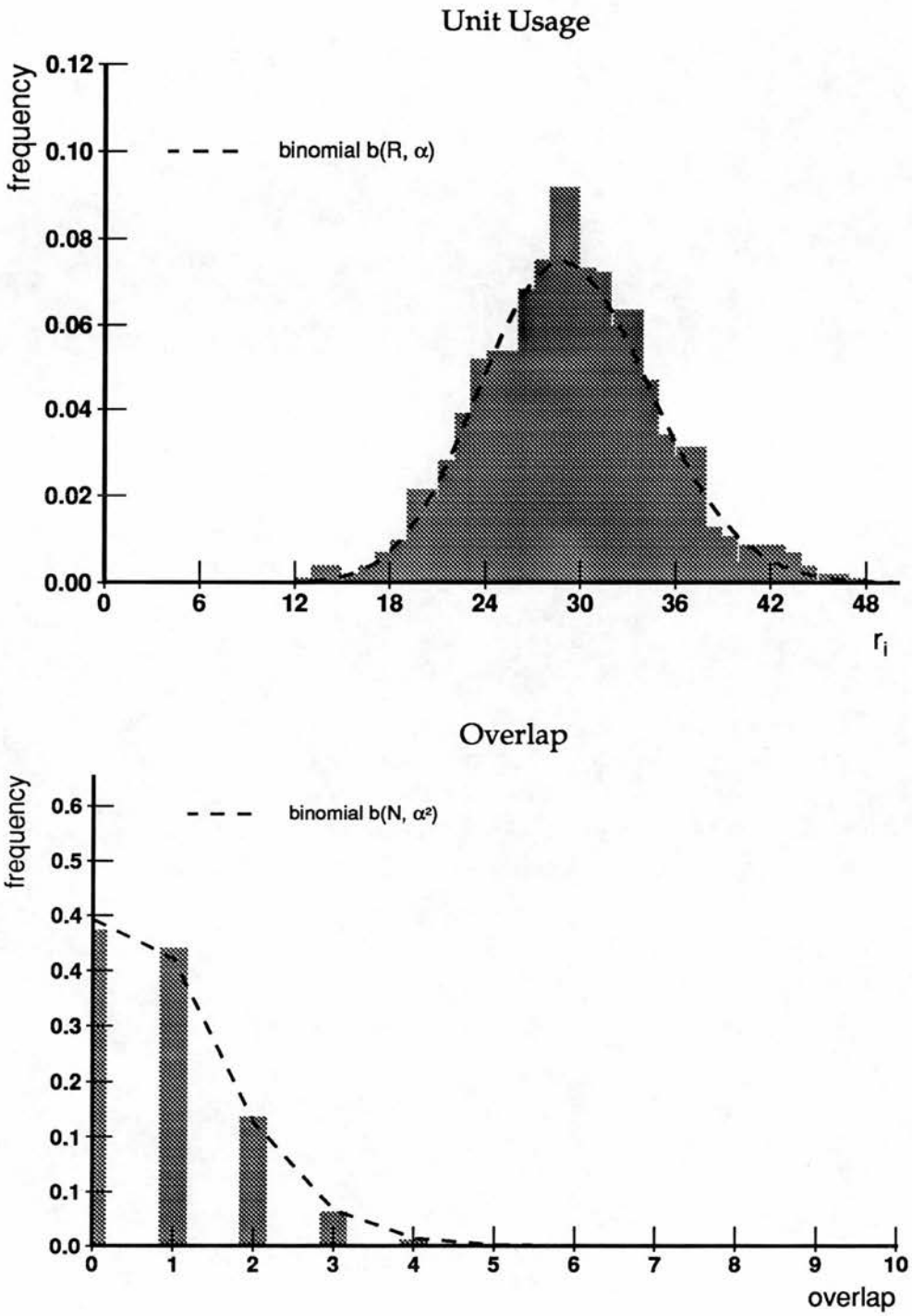

Figure A.4: Data for 1000 patterns from pattern set P3000.N1024.M30.1.input. N = 1024, M = 30.

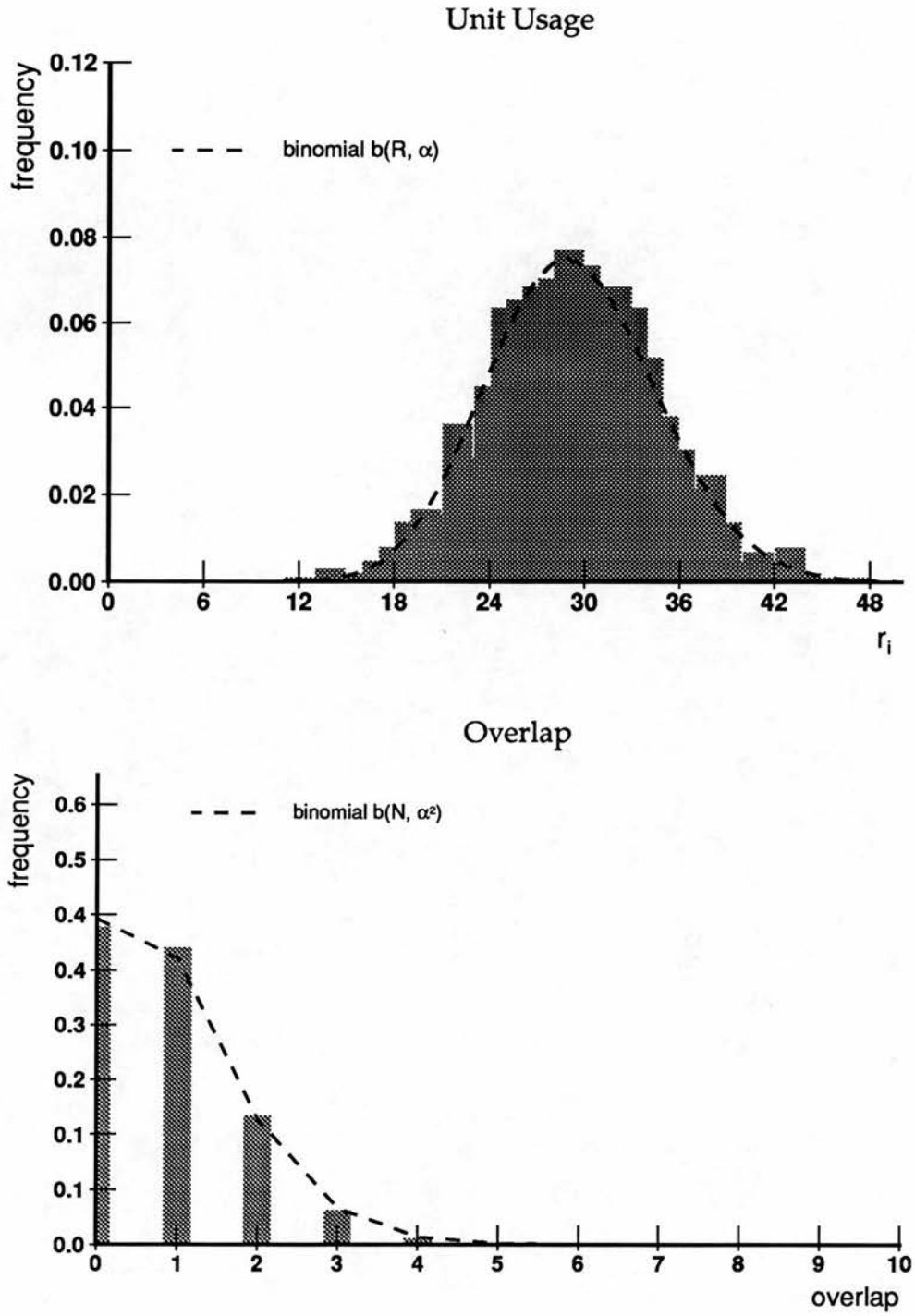

Figure A.5: Data for 1000 patterns from pattern set P3000.N1024.M30.5.input. N = 1024, M = 30.

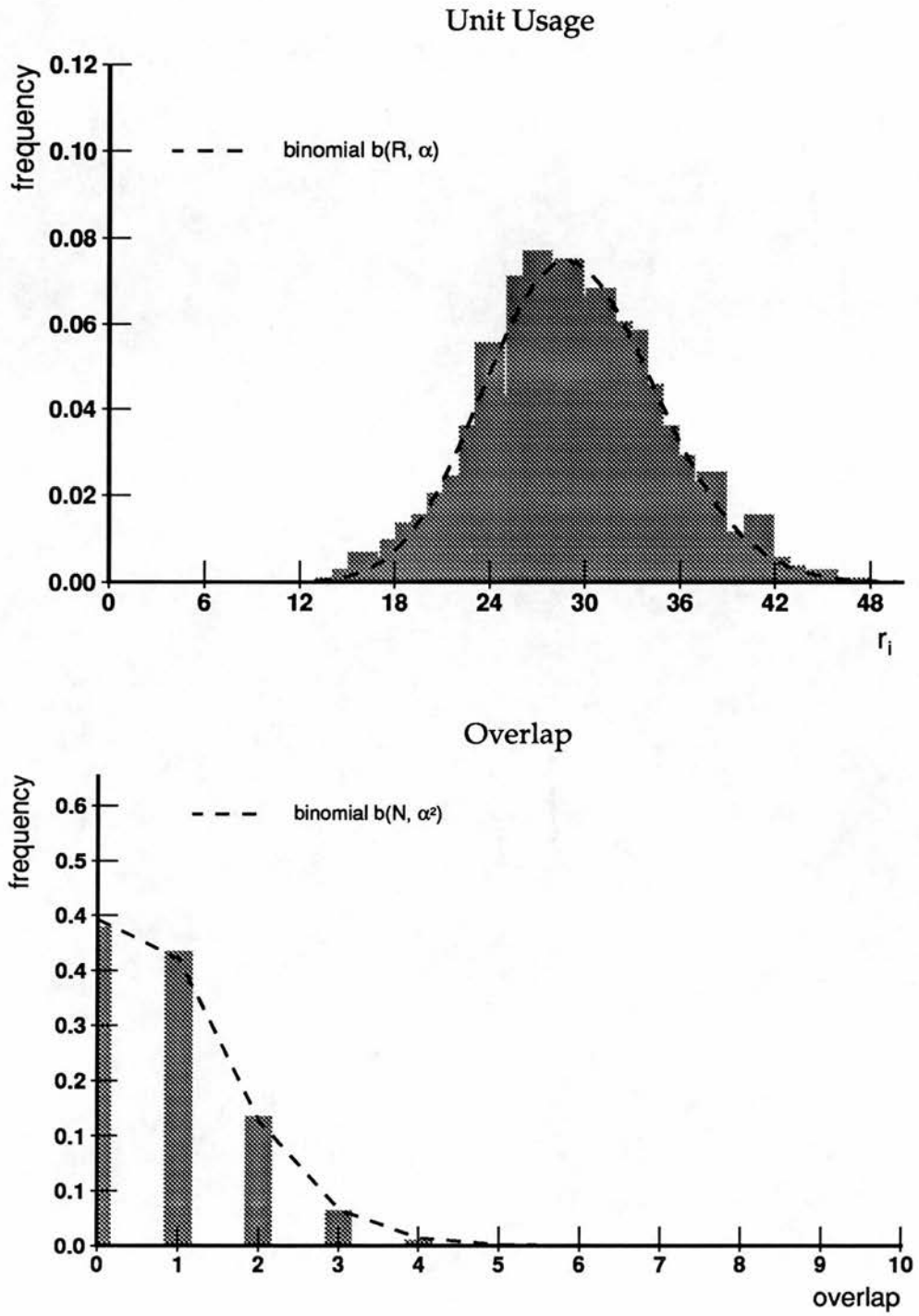

Figure A.6: Data for 1000 patterns from pattern set P3000.N1024.M30.8.input. N = 1024, M = 30.

Appendix B

What is a good cue?

What kind of recall performance do we want from these network memories? That depends on the use to which we will put the memory. We probably want as few bits error in the output patterns as possible. However, should we give equal weight to the false negatives and false positives? If we had to choose between having a genuine bit off or a spurious bit on, which would we choose?

In Marr (1970, 1971), the output of one associative net structure is used as the input to another. In the theory of the archicortex (1971), the output of the $\mathcal{P}_2$ layer is the input to the $\mathcal{P}_3$ layer and the output of $\mathcal{P}_3$ is an input to the neocortical model. One can imagine many other modular architectures where the output of one associative net is used as the input to another.

So, in this work, we **assume** that the output of the single layer associative net under study is to be used as input to another.

Thus, in order to evaluate on an output pattern, we ask what that pattern would be like as a recall cue to another net in which the target is one of the stored input patterns. With this in mind, let us examine the effects of using partial or noisy versions of stored input patterns as recall cues.

B.1 An initial view of recall errors as a function of missing and spurious bits in the cue

The quality of a recall cue can be defined in terms of the number of bits that are off in the cue that were on in the stored input pattern (missing bits) and the number of bits that are on in the cue that were off in the pattern (spurious). The question in the context of deciding the goodness or quality of a cue is what kind of output pattern does the net produce when presented with that cue. In the end, we want the number of bits wrong in the output pattern to be as small as possible and so at this point mean hamming distance is used as the error measure.

Consider the simulation results presented in Figure 3.7. These simulation results will be used for illustrative purposes in the discussion of cue quality measures. Analytic statements about the measure which is finally adopted in most of the thesis are made in the treatment of threshold setting strategies.

B.2 On measures of cue quality

It would be useful to express the quality of a cue by a single number. Since in general a cue contains a number of missing bits and a number of spurious bits, the object is to find a mapping of these two variables onto a single value that increases monotonically with the output errors elicited. A number of possible measures exists. stored. Also, a measure which is a natural part of analytic statements which describe the net would be very convenient. Several measures are discussed here. For each, how it changes as a function of the number of missing and spurious bits is presented and recall performance as a function of this measure is plotted for all of the cues used in the simulations described above. The measures addressed are hamming distance, cosine, an information theoretic quality measure used by Gardner-Medwin, Marr's genuine-to-spurious ratio, signal-to-noise, and a measure of the overlap of the dendritic sum distributions similar in spirit to signal-to-noise.

B.2.1 Hamming distance

The hamming distance between a cue and the pattern it is associated with is simply the number of missing genuine bits plus the number of spurious bits: $\Delta_{input} = \delta_g + m_s$. As a function of the missing and spurious bits in a cue it is simply a plane (see Figure B.1); From Figure 3.8, it can be seen that for low output error, cues with the same hamming distance from their input pattern elicit nearly the same output error. For larger output errors, cue hamming distance does not faithfully describe the kind of output errors which will arise. In Figure B.2, the results of the simulation shown in Figure 3.7 are replotted with cue hamming distance on the horizontal axis and mean output error on the vertical for each cue configuration. Since the scattergram of points alone does not provide information about what kind of cues elicited them, lines which join points corresponding to cues with fixed values of missing bits are also drawn for several values of δ_g. Note that when cue hamming distance is between approximately 60 and 160, the curves describing output error for lower values of missing bits are above those for higher values. Thus, at a fixed cue hamming distance, the cues with more spurious bits elicit more output errors. However, for large (fixed) values of cue hamming distance, greater numbers of missing bits elicit greater output errors. Informally, as long as no spurious bits are added to the cue, good recall can be obtained over quite a range of the number of missing bits. However, recall errors increase dramatically when spurious bits are added to cues with relatively few genuine bits.

B.2.2 Cosine

The cosine of the angle between the cue and an input pattern is an obvious measure of their similarity. Noting the expression for cosine:

$$\cos(\mathrm{cue}, \mathrm{pattern}) = \frac{\mathrm{cue} \cdot \mathrm{pattern}}{\|\mathrm{cue}\| \|\mathrm{pattern}\|}$$

we see that it has some of the properties we desire. For partial cues, the cosine decreases as $\sqrt{m_g}$. If noise is added to a pattern, the cosine, $\frac{m_g}{\sqrt{(m_g+m_s)M}}$,

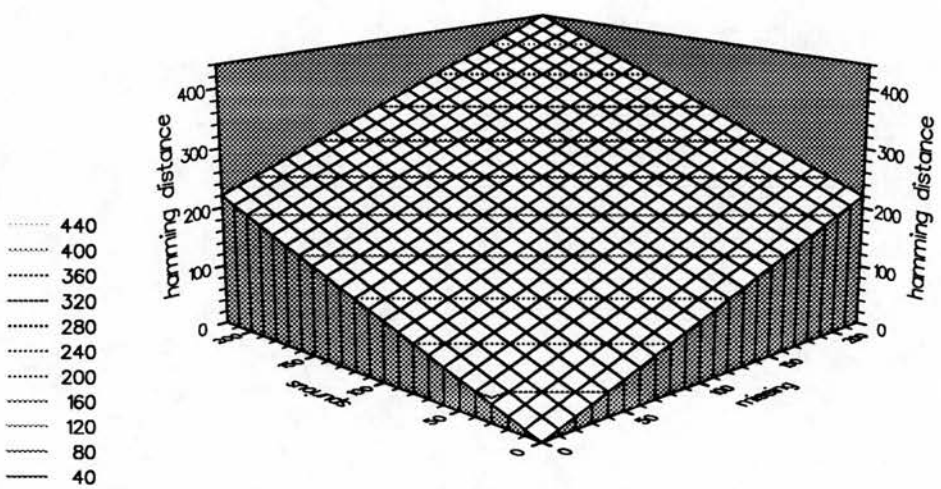

Figure B.1: Hamming distance between a cue and its corresponding stored input pattern as a function of the number of missing and spurious bits in the cue.

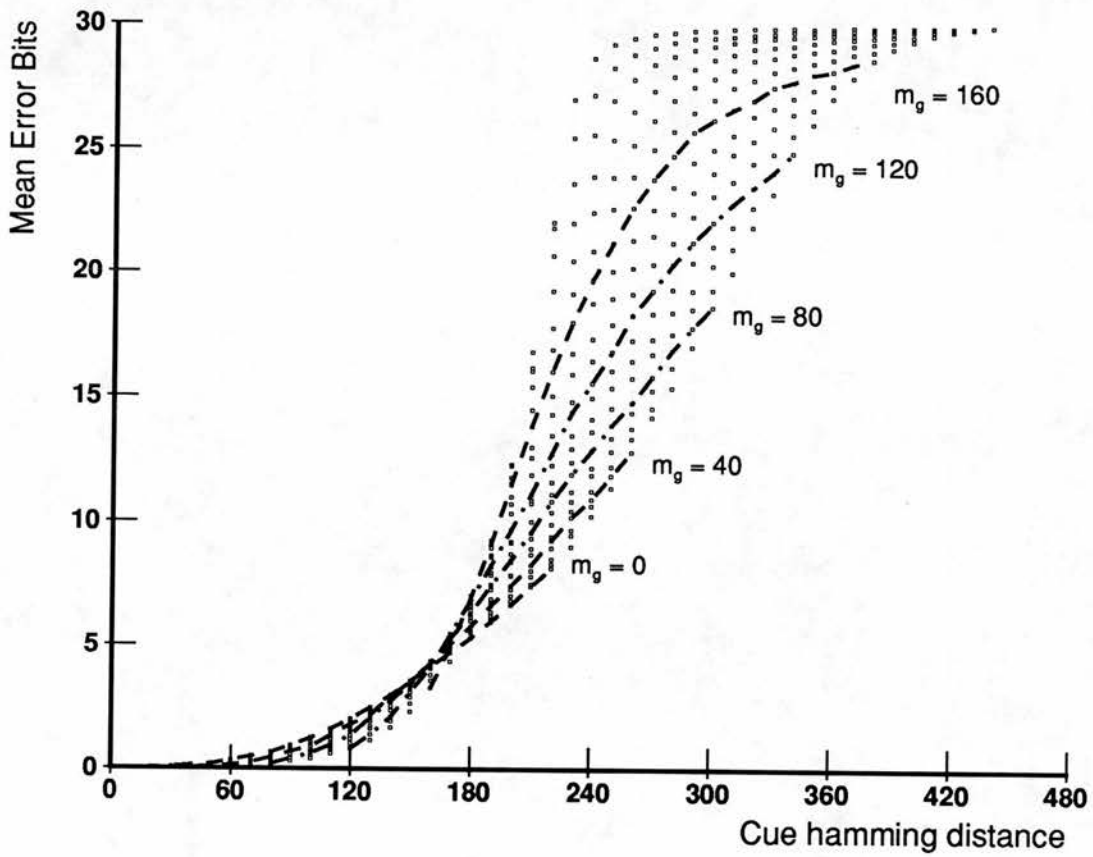

Figure B.2: Mean recall error as a function of cue hamming distance. Mean output error is plotted as a function of the hamming distance of the cue from its associated input pattern. Dashed lines join the points corresponding to cues with fixed number of missing bits.

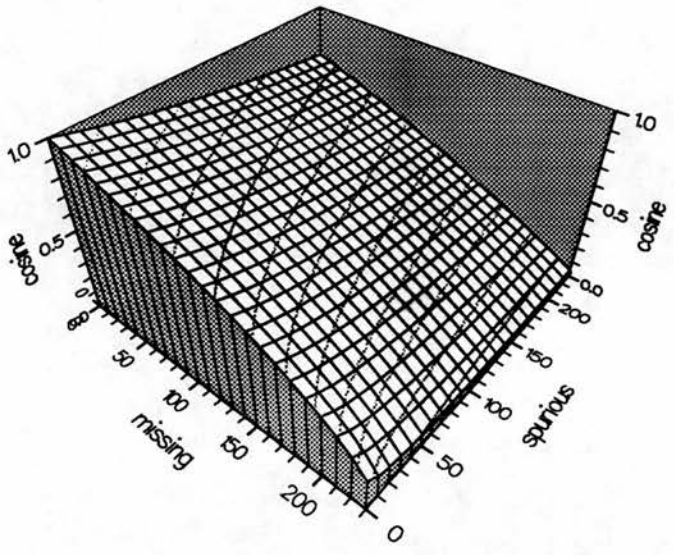

Figure B.3: Cosine of the angle between a cue a stored input pattern as a function of missing and spurious bits in the cue. $N_A = 8000$, $M_A = 240$. The missing bits increases along the x-axis, spurious bits along the y-axis. $\delta_g, m_g \in [0, 230]$.

decreases as m_s increases. Figure B.3 gives an overall view of cosine as a function of missing and spurious bits in the cue. As before, this is a 3 dimensional graph with the number of missing bits on the x axis, the number of spurious bits on the y axis, and $\cos(\mathrm{cue}, \mathrm{pattern})$ on the z axis. Interestingly, this view shows that the cosine surface is relatively flat and for a fixed number of missing bits as the number of spurious bits increases, cosine increases, but not steeply.

Does cosine faithfully reflect output errors? In Figure B.4, output error (from simulations) is plotted as a function of the cosine between a cue and its associated input pattern. The spread of output error for cues with the same cosine is quite wide; it is only tight when the cosine is greater than about 0.9. Lines joining points representing two types of cues are also drawn: one for partial cues (ones with no spurious bits) and one for noisy cues with equal numbers of missing and spurious bits. Not surprisingly, the recall performance using noisy cues is significantly worse than than that obtained using partial cues.

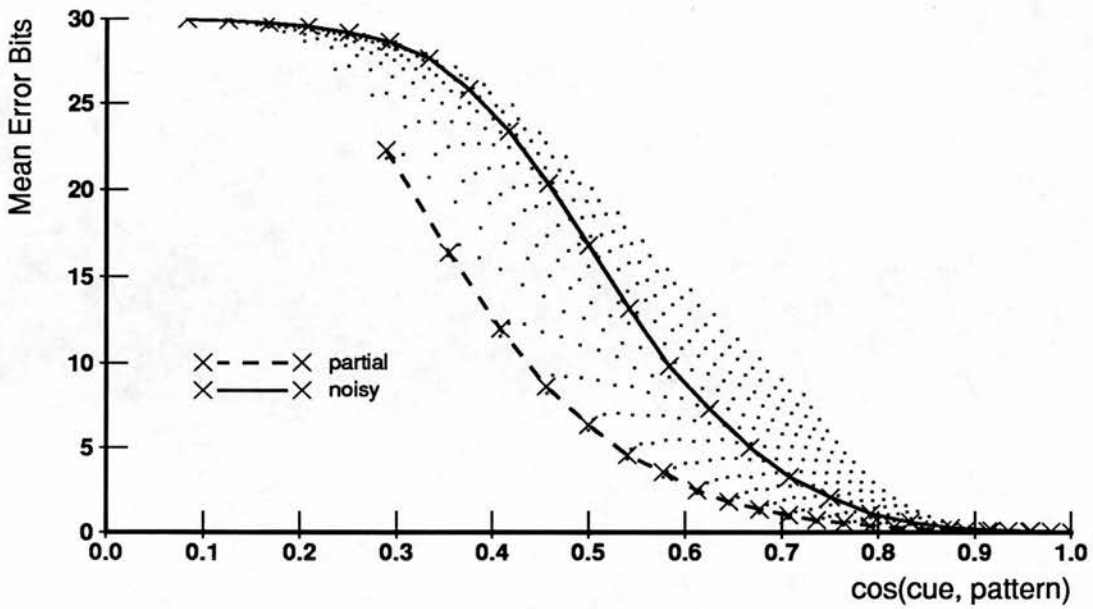

Figure B.4: Recall errors as a function of the cosine of the angle between the cue and the stored input pattern. Data are from simulations using the canonical parameter set, R = 1000, and 'Best T' thresholds.

B.2.3 Information theoretic quality

Gardner-Medwin (1989) develops an information theoretic measure of the quality of a pattern. Given an output pattern and a target pattern, this measure takes into consideration the information required to correct the pattern so that it matches the target and the information required to specify the target from scratch. From Gardner-Medwin (1989), the information required to specify the target pattern from scratch is given by

$$I_0 = NH(\alpha)$$

where

$$H(\alpha) = -\alpha \log_2(\alpha) - (1-\alpha)\log_2(1-\alpha) \quad \text{bits.}$$

The information required to specify the changes required so that the pattern matches the target is composed of two parts: the information required to identify the missing genuine bits and that required to indentify the spurious ones.

This is given by

$$I_c = mH(m_s/m) + (N - m)H(\delta_g/(N - m))$$

where $m = m_{cue}$. If I_c is small compared to I_0, the quality of the pattern is high, but if it close to I_0 the quality is low. So Gardner-Medwin defines the quality, Q, of a pattern to be

$$Q = 1 - \frac{I_c}{I_0}$$

Figure B.5 shows Q as a function of missing and spurious bits in a cue. Q is high when the number of missing and spurious bits is low. As the number of missing bits increases, quality decreases nearly, but not quite, linearly. For a fixed number of missing bits, quality also decreases as spurious bits are added, but this decrease is not dramatic.

As was done for hamming distance and cosine, recall errors are plotted as a function of cue quality in Figure B.6; performance is indeed quite different for cues with different configurations of missing and spurious bits which map to the same cue quality. As with the cosine measure, performance is much better using partial rather than noisy cues.

B.2.4 Genuine to spurious ratio

Marr (1971) was also interested in maximizing the number of genuine bits in an output pattern and minimizing the number of spurious ones. One of the performance measures he employed was the ratio of the number of genuine active units to the number of spurious ones in a pattern, m_g/m_s. Given a cue with a certain genuine-to-spurious ratio, he was interested in the genuine-to-spurious ratio in the output pattern elicited. If the ratio in the output pattern is greater than that of the input, the recall process in the net has provided a 'better' pattern than it was given. For a particular net, Marr defined the statistical threshold, **st**, to be the value of the ratio m_g/m_s such that cues whose genuine-to-spurious ratio was greater than **st** would tend to elicit output patterns whose genuine-to-spurious ratio was greater than that of the input and cues whose genuine-to-spurious ratio was less than **st** would tend to elicit outputs whose

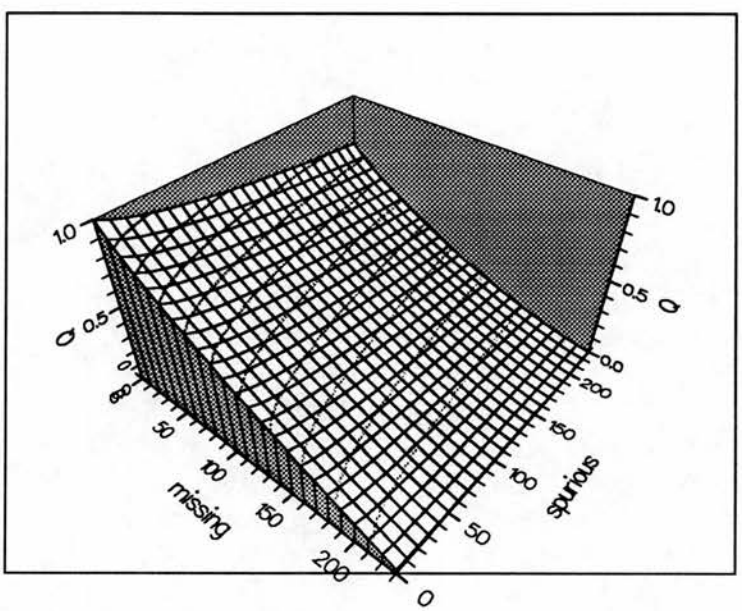

Figure B.5: Quality of a cue with respect to a pattern as a function of missing and spurious bits in the cue. $N_A = 8000$, $M_A = 240$.

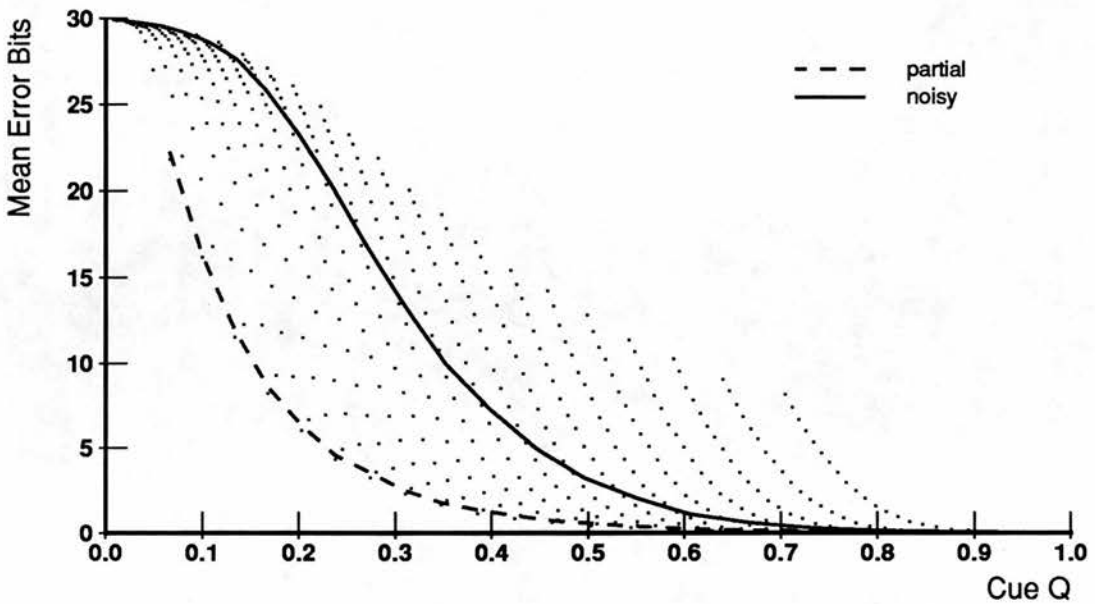

Figure B.6: Recall errors as a function of cue quality. Data are from simulations using the canonical parameter set, R = 1000, and 'Best T' thresholds.

ratio was less than that of the input. Thus the genuine-to-spurious ratio and the statistical threshold are functions of the parameters of the net and the thresholding strategy and not just of the cue and its associated input pattern. Marr used numerical methods to approximate the numbers of genuine and spurious bits in an output pattern for a cue defined by a given number of genuine and spurious bits. For this illustration, the simulation results are used.

Marr concentrated on the case where the number of active bits in a cue was constant. This is the same as the $m_g + m_s = M_A$ case often plotted in this thesis. As m_s goes to zero, the ratio m_g/m_s goes to infinity which limits its usefulness with respect to partial patterns. However, Marr focussed on that middle ground of noisy patterns that tended to elicit patterns which were about as noisy, so this did not concern him. Since recall from noisy patterns is more difficult than recall from partial ones, the noisy case is more informative with respect to pushing the capabilities of the net. For noisy cues, the statistical threshold provides a succinct measure of the recall capabilities of a particular net after a fixed number of patterns have been stored in it. However, this thesis is concerned with partial cues as well so the genuine-to-spurious ratio on its

own will not suffice as a performance measure in this work. Also, it does not capture the number of units active. In a cue it may be possible to make this constant, but the thresholding strategy used in the net may not guarantee a constant number of active units in the output. For instance, an output pattern may have a high genuine-to-spurious ratio, but the number of genuine bits may be low, in which case it would be difficult to classify it as a 'good' pattern for the reason illustrated above - patterns with low numbers of genuine bits (with respect to some stored input pattern) do not elicit good recall when used as cues.

In order to provide a comparison with the other quality measures under consideration, the m_g/m_s ratio is plotted as a function of δ_g and m_s in Figure B.7 and performance as a function of m_g/m_s for (almost) partial and noisy cues in Figure B.8. In Figure B.7, m_s goes from 10 to 220 instead of from 0 to 220 as it does in the other 3 dimensional graphs which relate some cue quality measure to δ_g and m_s; in Figure B.8, the 'partial' cues have 10 noise bits added. This genuine-to-spurious measure behaves quite simply: for fixed m_g it varies inversely with m_s and for fixed m_s it varies linearly with m_g. Figure B.8 shows cues with genuine-to-spurious ratios above about 3.0 elicit low output errors, and as the ratio goes below that output errors increase dramatically. In this sense it seems to provide a good indicator of the recall behaviour of the net, though cues with the same genuine-to-spurious ratio do elicit a range of recall errors.

B.3 Measures of dendritic sum distribution overlap

B.3.1 Signal-to-noise ratio

The signal-to-noise ratio is one measure of the overlap of two distributions and so it should be related to the output errors observed. There are several signal-to-noise measures used by different workers. A common one is

$$\tau_1 = \frac{(\mu_g - \mu_s)^2}{(\sigma_g^2 + \sigma_s^2)/2}$$

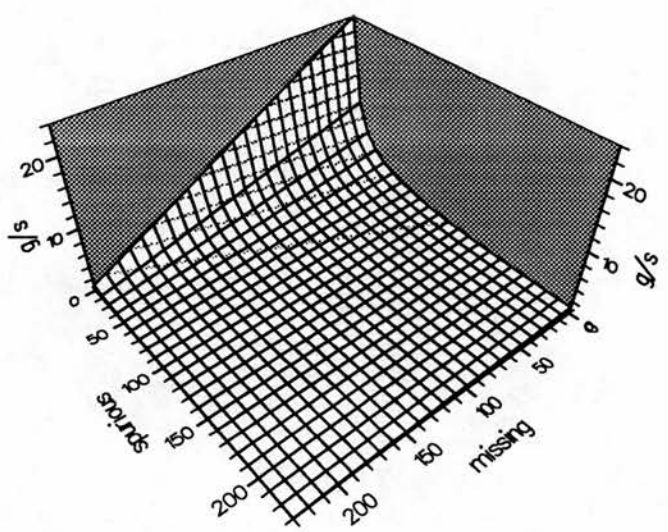

Figure B.7: Genuine-to-spurious ratio of a cue with respect to a pattern as a function of missing and spurious bits in the cue. $N_A = 8000$, $M_A = 240$. Genuine-to-spurious ratio is on the vertical axis. $z \in [0, 24]$.

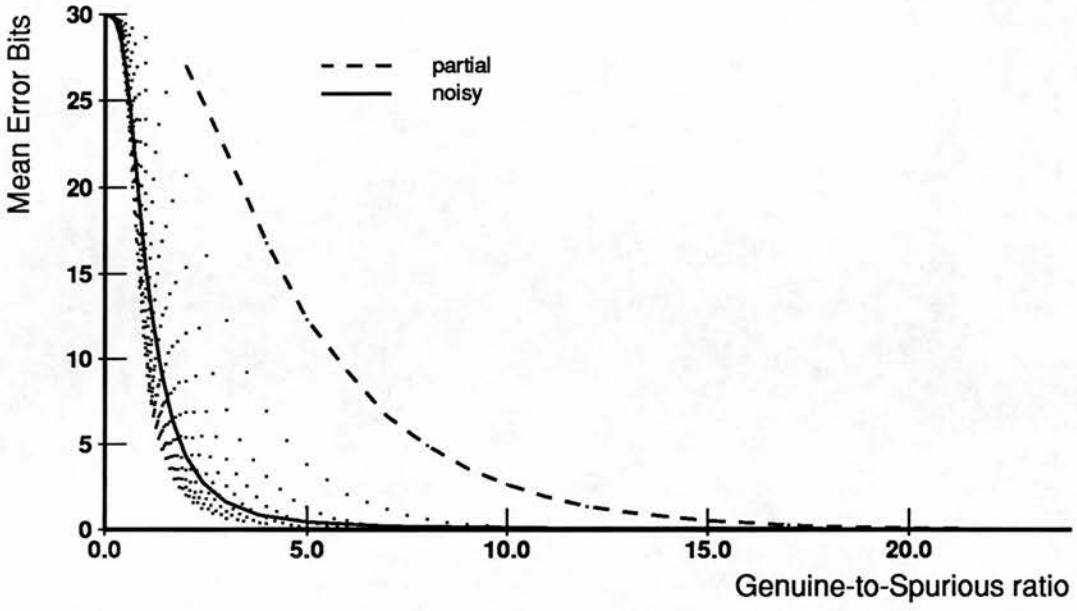

Figure B.8: Recall errors as a function of the genuine-to-spurious ratio of the cue with respect to the input pattern. Data are from simulations using the canonical parameter set, R = 1000, and 'Best T' thresholds.

where μ and σ are respectively the mean and standard deviation of a distribution. This measure is appropriate when both distributions are normal and their standard deviations are nearly equal, but it does not give an faithful measure of the overlap of the distributions if the standard deviations are quite different. It also turns out that partial or noisy cues which give rise to the same signal-to-noise elicit very different recall errors in simulations.

Willshaw (personal communication) has suggested another expression for signal-to-noise which is more applicable when the standard deviations of the low and high distributions are quite different. This is

$$\tau_2 = \frac{(\mu_g - \mu_s)^2}{(\sigma_g^2 + \sigma_s^2 - \sigma_g \sigma_s)}$$

When the standard deviations are equal, it is the same as the common expression for signal-to-noise, but when one of the variances approaches zero the denominator approaches the other variance instead of the average.

Cues with low numbers of missing and spurious bits have a high signal-to-

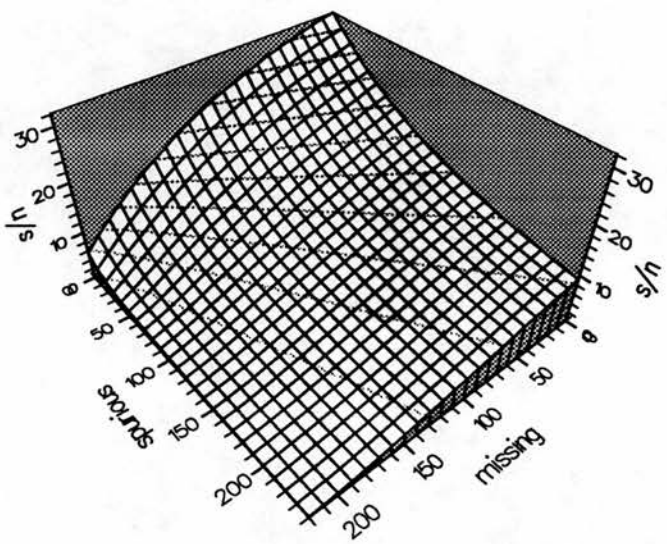

Figure B.9: Modifed Signal-to-noise ratio of a cue as a function of missing and spurious bits in the cue. $N_A = 8000$, $M_A = 240$. Signal-to-noise is on the vertical axis.

noise ratio and those with many missing or spurious bits map to a low ratio. Figure B.9 shows that this measure decreases steeply as the missing and/or spurious bits increase.

It turns out that this measure is a reasonable predictor of output errors. Figure B.10 shows mean output errors (from the simulations) as a function of this signal-to-noise, τ_2. Different combinations of missing and spurious bits which map to the same signal-to-noise elicit somewhat different errors, but the the spread is less than that observed in the graphs relating some of the other cue quality measures to mean output error. For cues which elicit low output errors, this signal-to-noise measure is a consistent predictor of error.

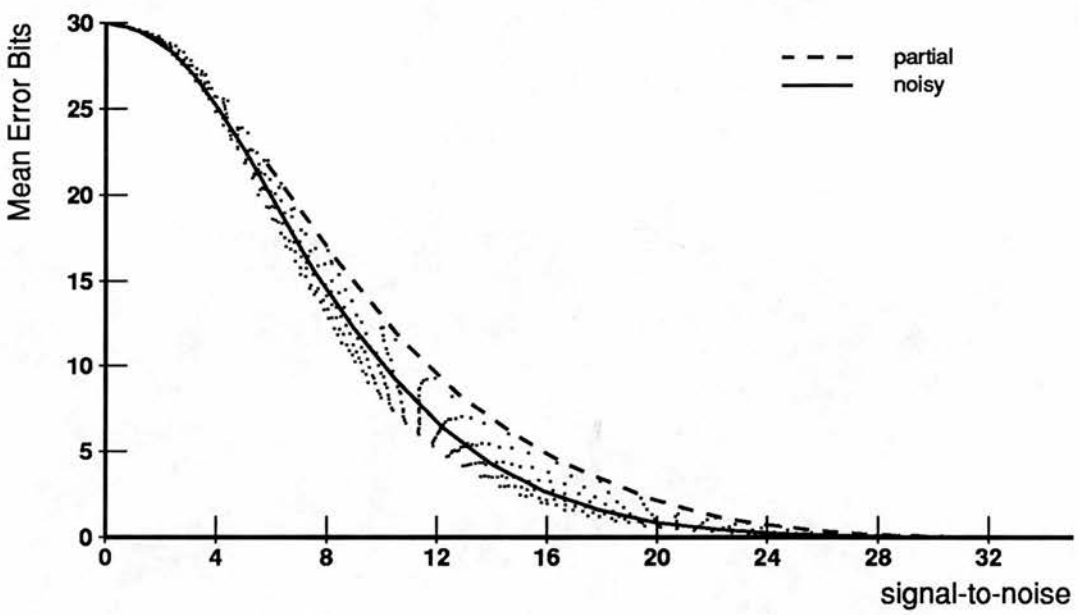

Figure B.10: Comparing recall errors using partial or noisy patterns as cues as a function of the signal-to-noise ratio τ_2 of the dendritic sum distributions. Data are from simulations using the canonical parameter set, R = 1000, and 'Best T' thresholds.

B.3.2 Comparing dendritic sum distribution overlap to performance

Using thresholding strategies based on dendritic sums, recall errors increase as the d_s and d_g distributions overlap more. The signal-to-noise ratio provides one measure of overlap which does a reasonable job of predicting output errors, but can we do better? Let us consider the overlap of the distributions, which is just one step away from output errors.

For the present discussion let us use the area under the overlap of the two distributions as a measure of overlap. That is,

$$\text{overlap} = \sum_{x=0}^{M_A} \min(P(d_s = x), P(d_g = x))$$

The overlap for various values of missing and spurious bits was calculated from the expressions for the d_s and d_g probability distributions and plotted in Figure B.11 as a function of δ_g and m_s. As δ_g and m_s increase, so does the overlap of the d_s and d_g distributions. When δ_g and m_s are both low, overlap is low and the surface relatively flat, but it rises sharply for noisy cues. It also increases for strictly partial cues as the number of missing bits increses, but it does so much more slowly than it does for noisy cues. Using the data from Figure B.4, a scattergram of recall errors as a function of distribution overlap is plotted in Figure B.12. We see that patterns with the same overlap have nearly the same recall errors. This is not surprising since this measure of overlap and the thresholding strategy used in the simulations are both functions of the dendritic sum distributions.

Thus the overlap of the dendritic sum distributions seems to capture the relative 'cost' of false positive and false negative bits in a cue, which is what was desired. Unlike hamming distance or Gardner-Medwin's Q, it is not a function solely of two patterns – it is computed as a function of the parameters of the network under study. Thus it is not as general-purpose as hamming distance or Q. It was stressed earlier that the question of how good a pattern is depends on ones purpose and this measure reflects that thinking. Overlap directly reflects the 'goodness' of the pattern when used as a recall cue in the net being considered.

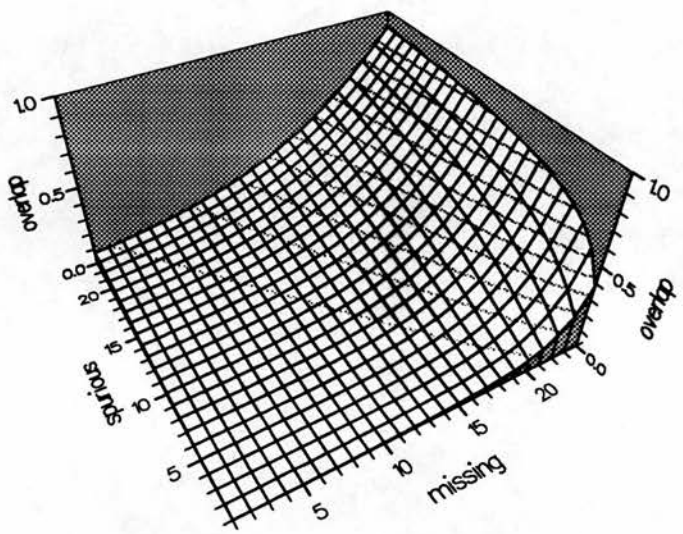

Figure B.11: Overlap of d_s and d_g distributions as a function of missing and spurious bits in the cue. The canonical parameter set was used in the calculations.

Overlap and signal-to-noise should be related. Figure B.13 plots the signal-to-noise ratio as a function of overlap for the usual cues (all the missing and spurious combinations). Large overlap corresponds to small signal-to-noise. However, cues with the same overlap may have somewhat different signal-to-noise ratios; the data points in the scattergram follow a curve, but do not all lie on it. Replotting using a logarithmic x axis shows that signal-to-noise is nearly linearly related to the log of overlap (Figure B.14). Overlap faithfully predicts mean output errors. The dispersion in the relationship between signal-to-noise and overlap corresponds to the dispersion observed error data as a function of signal-to-noise (Figure B.10).

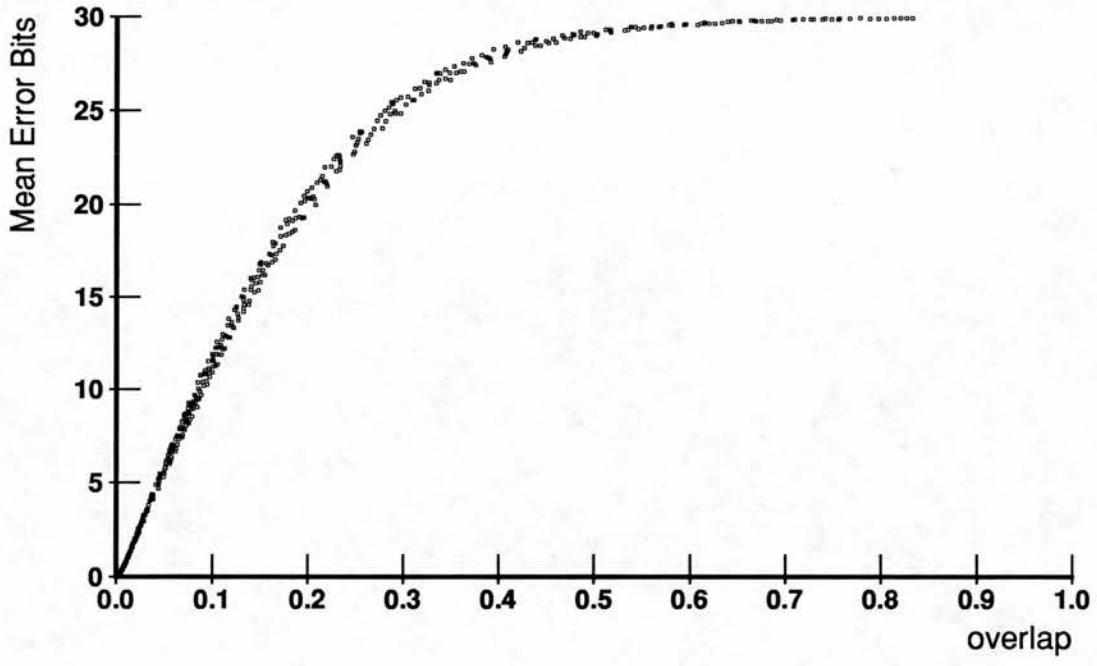

Figure B.12: Recall errors as a function of the overlap of the dendritic sum distributions corresponding to the cue. Data are from simulations using the canonical parameter set, R = 1000, and 'Best T' thresholds.

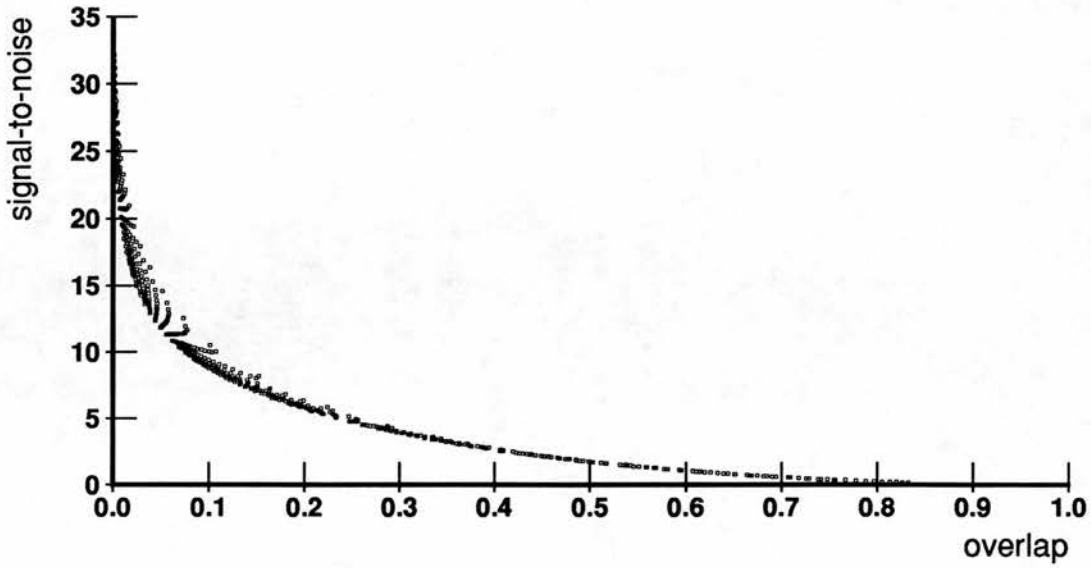

Figure B.13: Comparing signal-to-noise ratio using cues with many combinations of missing and spurious bit as a function of the overlap of the dendritic sum distributions corresponding to the cue (for nets defined by the canonical parameters).

B.4 Discussion of cue quality measures

A number of measures which relate a cue to a pattern have been illustrated. Of these, hamming distance, cosine, and information-theoretic quality are functions of just the two vectors, while the others are functions of the parameters of a network as well. Marr's genuine-to-spurious ratio also depends on the thresholding strategy. Since the quality of a cue is judged with respect to the output errors it elicits when presented to a net, it makes sense that the parameters of the net are required to determine cue quality. However, a measure that is a simply a metric between vectors is more appealing. The overlap of the dendritic sum distributions faithfully reflects recall errors, but then it should since it is but one step away from them with the thresholding strategy employed in the simulations used for these illustrations. For most of the measures, different cues with the same value of the measure give rise to different mean output error. When output error is relatively low, hamming distance does a fair job of reflecting output error, so is not a bad measure of the goodness of a cue.

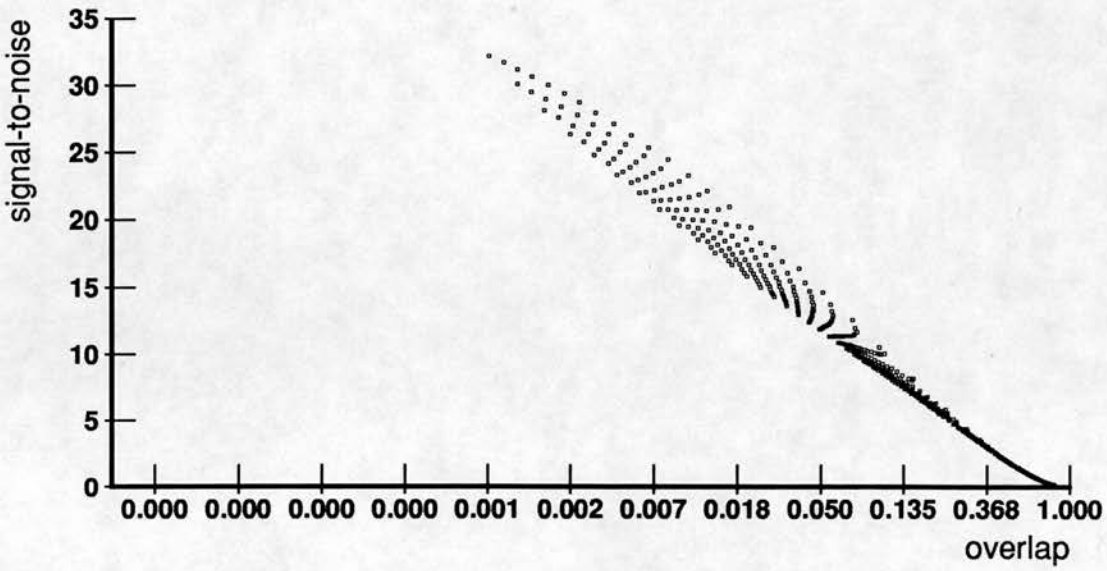

Figure B.14: Comparing signal-to-noise ratio using cues with many combinations of missing and spurious bit as a function of ln(overlap) of the dendritic sum distributions corresponding to the cue (for nets defined by the canonical parameters).

In order to understand the behaviour of these network architectures, it it useful to be able to make analytic statements about them where possible. Thus the measure employed should be a natural part of or conveniently fall out of these analytic statements. Using the expressions for the dendritic sum distributions developed earlier in this thesis, the expected number of genuine and spurious output units can be calculated for any threshold setting. The number of genuine and spurious bits in the output pattern can be used to calculate hamming distance, cosine, or information-theoretic quality of the output. Hamming distance is the natural error measure to employ particularly since it is used by a number of other workers, and results stated in terms of it can be readily compared with other work.

Bibliography

Amaral, DG, Ishizuka, N, & Claiborne, B (1990). Neurons, numbers and the hippocampal network. In: *Progress in Brain Research*, (J Storm-Mathisen, J Zimmer, & OP Ottersen, eds) volume 83 chapter 1, pp. 1–11. Elsevier.

Andersen, P, Bliss, TVP, & Skrede, K (1971). Lamellar organization of hippocampal excitatory pathways. *Exp Brain Res*, **13**, 222–238.

Bliss, TVP & Gardner-Medwin, AR (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the unanaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology (London)*, **232**, 357–374.

Bliss, TVP & Lømo, T (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology (London)*, **232**, 331–356.

Boss, BD, Peterson, GM, & Cowan, WM (1985). On the number of neurons in the dentate gyrus of the rat. *Brain Res*, **338**, 144–150.

Boss, BD, Turlejski, K, Stanfield, BB, & Cowan, WM (1987). On the number of neurons in fields ca1 and ca3 of the hippocampus of srague-dawley and wistar rats. *Brain Res*, **406**, 280–287.

Brindley, GS (1967). The classification of modifiable synapses and their use in models for conditioning. *Proceedings of the Royal Society of London, Series B*, **168**, 361–376.

Eichenbaum, H, Fagan, A, Mathews, P, & Cohen, NJ (1988). Hippocampal system dysfunction and odor discrimination: Impairment or facilitation depending on cognitive strategies. *Behavioral Neuroscience*, **102**, 331–339.

Friedlander, MJ, Sayer, RJ, & Redman, SJ (1990). Evaluation of long-term potentiation of small compound and unitary epsps at the hippocampal ca3/ca1 synapse. *J Neuroscience,* **10**, 814–825.

Gardner-Medwin, AR (1976). The recall of events through the learning of associations between their parts. *Proceedings of the Royal Society of London, Series B,* **194**, 375–402.

Gardner-Medwin, AR (1990). Constraints on the use of inhibition in neural discrimination of learned contexts. *Journal of Physiology (London),* **430**, 27–28.

Gray, JA (1982). *The neuropsychology of anxiety*. Oxford: Oxford University Press.

Green, EJ, McNaughton, BL, & Barnes, CA (1990). Exploration-dependent modulation of evoked responses in fascia dentata: Dissociation of motor, EEG, and sensory factors and evidence for a synaptic efficacy change. *The Journal of Neuroscience,* **10** (5), 1455–1471.

Grossberg, S (1988). Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Networks,* **1**, 17–61.

Hebb, DO (1949). *The Organisation of Behaviour*. New York: Wiley.

Lømo, T (1971). Potentiation of monosynaptic epsp's in the perforant path – dentate granule cell synapse. *Expl Brain Res,* **12**, 46–63.

Lorente de No, R (1934). Studies on the structure of the cerebral cortex. ii. continuation of the study of the ammonic system. *J Psychol Neurol,* **46**, 113–177.

Marr, D (1969). A theory of cerebellar cortex. *Journal of Physiology (London),* **202**, 437–470.

Marr, D (1970). A theory for cerebral cortex. *Philosophical Transactions of the Royal Society of London, Series B,* **176**, 161–264.

Marr, D (1971). Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society of London, Series B,* **262**, 23–81.

Marr, D (1982). *Vision*. San Francisco: Freeman.

McNaughton, BL (1989). Neuronal mechanisms for spatial computation and information storage. In: *Neural Connections and Mental Computation*, (L Nadel, LA Cooper, P Culicover, & RM Harnish, eds). MIT Press, Bradford Books Cambridge, MA.

Morris, RGM (1983). An attempt to dissociate 'spatial-mapping' and 'working-memory' theories of hippocampal function. In: *Neurobiology of the hippocampus*, (W Siefert, ed). Academic Press London.

Morris, RGM, Garrud, P, Rawlins, JNP, & O'Keefe, J (1982). Place navigation impaired in rats with hippocampal lesions. *Nature*, **297**, 681–683.

O'Keefe, J (1976). Place units in the hippocampus of the freely moving rat. *Exp Neurol*, **51**, 78–109.

O'Keefe, J & Dostrovsky, J (1971). The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain Res*, **34**, 171–175.

Olton, DS & Samuelson, RJ (1976). Remembrance of places passed: spatial memory in rats. *J Exp Psych Animal Behav Proc*, **2**, 97–116.

Palm, G (1981). On the storage capacity of an associative memory with randomly distributed storage elements. *Biological Cybernetics*, **39**, 125–127.

Rawlins, JNP (1985). Associations across time: The hippocampus as a temporary memory store. *The Behavioural and Brain Sciences*, **8**, 479–496.

Rolls, ET (1989). Parallel distributed processing in the brain: implications of the functional architecture of neuronal networks in the hippocampus. In: *Parallel Distributed Processing*, (RGM Morris, ed) pp. 286–308. Oxford University Press Oxford.

Rosene, DL & Van Hoesen, GW (1987). The hippocampal formation of the primate brain. In: *Cerebral Cortex*, (EG Jones & A Peters, eds) volume 6. Plenum.

Rumelhart, D, McClelland, J, & the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. Cambridge: MIT Press.

Rumelhart, D & Zipser, D (1985). Feature discovery by competitive learning. *Cognitive Science,* **9**, 75–112. Reprinted in (Rumelhart *et al.*, 1986, chapter 5).

Scoville, WB & Milner, B (1957). Loss of recent memory after bilateral hippocampal lesions. *J Neurol Neurosurg Psychiat,* **20**, 11–12.

Shannon, CE (1963). The mathematical theory of communication. In: *The mathematical theory of communication,* (CE Shannon & W Weaver, eds). University of Illinois Press Urbana.

Squire, L, Shimamura, AP, & Amaral, DG (1989). Memory and the hippocampus. In: *Neural Models of Plasticity,* (J Byrne & W Barry, eds). Academic Press New York.

Steinbuch, K (1961). Die lernmatrix. *Kybernetik,* **1**, 36–45.

Thompson, LT & Best, PJ (1989). Place cells and silent cells in the hippocampus of freely-behaving rats. *The Journal of Neuroscience,* **9** (7), 2382–2390.

Willshaw, DJ (1971). *Models of Distributed Associative Memory.* PhD thesis University of Edinburgh Edinburgh.

Willshaw, DJ & Buckingham, JT (1990). An assessment of marr's theory of the hippocampus as a temporary memory store. *Philosophical Transactions of the Royal Society of London, Series B,* **329**, 205–215.

Willshaw, DJ, Buneman, OP, & Longuet-Higgins, HC (1969). Non-holographic associative memory. *Nature,* **222**, 960–962.

Willshaw, DJ & Dayan, P (1990). Optimal plasticity in matrix memories: What goes up MUST come down. *Neural Computation,* **2**, 85–93.

Zola-Morgan, S & Squire, L (1990). The primate hippocampal formation: evidence for a time-limited role in memory storage. *Science,* **250**, 288–290.